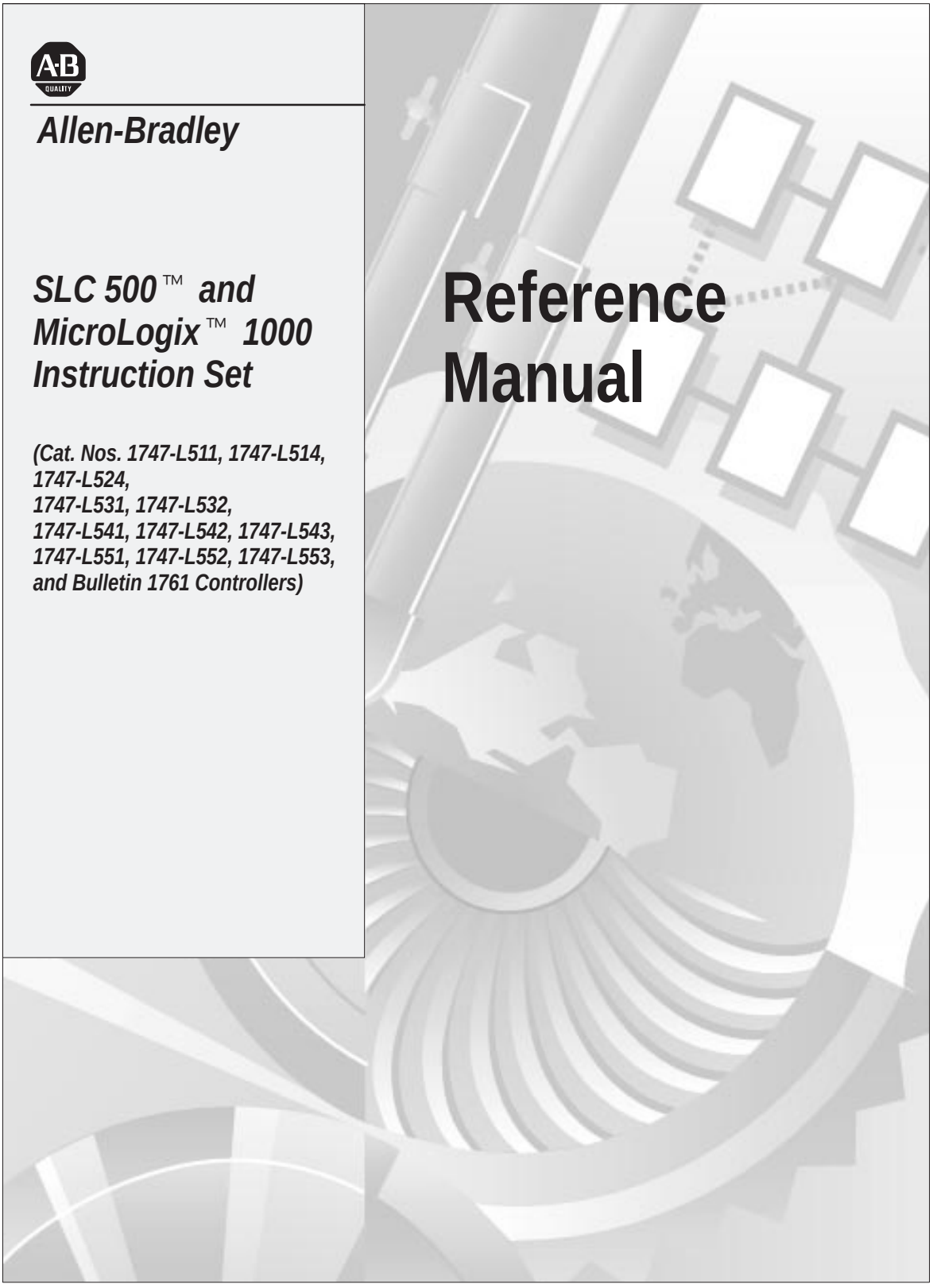


Allen-Bradley

SLC 500™ and MicroLogix™ 1000 Instruction Set

(Cat. Nos. 1747-L511, 1747-L514, 1747-L524, 1747-L531, 1747-L532, 1747-L541, 1747-L542, 1747-L543, 1747-L551, 1747-L552, 1747-L553, and Bulletin 1761 Controllers)

Reference Manual

Important User Information

Because of the variety of uses for the products described in this publication, those responsible for the application and use of this control equipment must satisfy themselves that all necessary steps have been taken to assure that each application and use meets all performance and safety requirements, including any applicable laws, regulations, codes, and standards.

The illustrations, charts, sample programs and layout examples shown in this guide are intended solely for purposes of example. Since there are many variables and requirements associated with any particular installation, Allen-Bradley does not assume responsibility or liability (to include intellectual property liability) for actual use based on the examples shown in this publication.

Allen-Bradley publication SGI-1.1, Safety Guidelines for the Application, Installation, and Maintenance of Solid-State Control (available from your local Allen-Bradley office), describes some important differences between solid-state equipment and electromechanical devices that should be taken into consideration when applying products such as those described in this publication.

Throughout this manual, we use notes to make you aware of safety considerations:

Identifies information about practices or circumstances that can lead to personal injury or death, property damage, or economic loss.

Attention statements help you to:

- identify a hazard
- avoid the hazard
- recognize the consequences

Note *Identifies information that is critical for successful application and understanding of the product.*

Summary of Changes

New Information Added to this Manual

The list below summarizes the changes that have been made to the Instruction Set Reference manual since the last printing:

For this New Information:	See:
Addition of new SLC 5/05 Processor.	throughout
Two new MicroLogix 1000 chapters: MicroLogix Communication Instruction MicroLogix Communication Protocols	 Ch. 9 Ch. 14
Revised information on how to use the status bits for the MSG instruction and new ladder logic examples.	Ch. 8
Revised Interrupt Latency specifications for STI, DII, and IOI.	App. D
New application example: Interfacing with Enhanced Bar Code Decoders Over DH-485 Network.	App. H

To help you find new information and updated information in this release of the manual, we have included change bars as shown to the right of this paragraph.

Table of Contents

Preface

Read this preface to familiarize yourself with the rest of the manual. It provides information concerning:

- who should use this manual
- purpose of this manual
- conventions used in this manual

Who Should Use this Manual

Use this manual if you are responsible for designing, installing, programming, or troubleshooting control systems that use Allen–Bradley small logic controllers.

You should have a basic understanding of SLC 500™ products. If you do not, contact your local Allen–Bradley representative for the proper training before using this product.

Purpose of this Manual

This manual is a reference guide for the SLC 500 processors and the MicroLogix 1000 controllers. This manual:

- provides status file functions
- provides the instructions used in your ladder logic programs
- complements the online help available at the terminal

Contents of this Manual

Chapter	Title	Contents
Appendix C	Memory Usage and Instruction Execution Times	Provides the user memory capacity and instruction execution times. Also describes how to estimate the total memory usage of a system.
Appendix D	Estimating Scan Time	Provides interrupt latency, M0/M1 access time information, and worksheets for estimating scan times.
Appendix E	Programming Instruction References	Provides a listing of instructions with their parameters and valid file types.
Appendix F	Data File Organization and Addressing	Provides details on data files, covering file formats and how to create and delete data.
Appendix G	Number Systems	Describes the hexadecimal, binary, and decimal numbering systems along with the floating point format.
Appendix H	Application Example Programs	Provides advanced application examples for the high-speed counter, sequencer, and bit shift instructions.

Related Documentation

The following documents contain additional information concerning Allen–Bradley SLC products. To obtain a copy, contact your local Allen–Bradley office or distributor.

For	Read this Document
An overview of the SLC 500 family of products	SLC 500 System Overview
An introduction to APS for first–time users, containing basic concepts but focusing on simple tasks and exercises, and allowing the reader to begin programming in the shortest time possible	APS Quick Start for New Users
A procedural and reference manual for technical personnel who use the APS import/export utility to convert APS files to ASCII and conversely ASCII to APS files	APS Import/Export User Manual
A training and quick reference guide to APS	SLC 500 Software Programmer's Quick Reference Guide, Publication Number ABT-1747-TSJ50—available on PASSPORT at a list price of $50.00
A guide of common procedures used in APS	SLC 500 Software Common Procedures Guide, Publication Number ABT-1747-TSJ50—available on PASSPORT at a list price of $50.00
A procedural manual for technical personnel who use APS to develop control applications	Rockwell Software Advanced Programming Software (APS) User Manual
A description on how to install and use your *Fixed* SLC 500 programmable controller	Installation and Operation Manual for Fixed Hardware Style Programmable Controllers, Catalog Number 1747-6.1

For	Read this Document
A description on how to install and use your *Modular* SLC 500 programmable controller	Installation and Operation Manual for Modular Hardware Style Programmable Controllers, Publication Number 1747-6.2
A description on how to install and use your MicroLogix 1000 controllers. This manual also contains status file data and instruction set information for the micro controllers.	MicroLogix 1000 Controllers User Manual, Publication Number 1761-6.3.
A complete listing of current Allen–Bradley documentation, including ordering instructions. Also indicates whether the documents are available on CD–ROM or in multi–languages.	Allen–Bradley Publication Index, Publication Number SD499
A glossary of industrial automation terms and abbreviations	Allen–Bradley Industrial Automation Glossary, Publication Number AG-7.1

Common Techniques Used in this Manual

The following conventions are used throughout this manual:

- Bulleted lists provide information, not procedural steps.
- Numbered lists provide sequential steps or hierarchical information.
- Text in `this font` indicates words or phrases you should type.
- *Italic* type is used for emphasis.

The following table summarizes the conventions used to distinguish the differences between the SLC 5/03, SLC 5/04, and SLC 5/05 keyswitch positions, the processor modes, and the actual display on the programmer status line.

When Referring to the Keyswitch Position	When Referring to the Processor Mode	When Referring to the Status Line
RUN position	Run mode	RUN
REMote position	Run mode	REM RUN
	Program mode	REM PROG
	Test – Single Step mode	REM SRG
	Test – Single Scan mode	REM SSN
	Test – Continuous Scan mode	REM CSN
PROGram position	Program mode	PROG

1 Basic Instructions

This chapter contains general information about the basic instructions and explains how they function in your application program. Each of the basic instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the basic instructions in use.

Bit Instructions

Instruction		Purpose	Page
Mnemonic	**Name**		
XIC	Examine if Closed	Examines a bit for an On condition.	**1–9**
XIO	Examine if Open	Examines a bit for an Off condition.	**1–9**
OTE	Output Energize	Turns a bit On or Off.	**1–10**
OTL and OTU	Output Latch and Output Unlatch	OTL turns a bit on when the rung is executed, and this bit retains its state when the rung is not executed or a power cycle occurs. OTU turns a bit off when the rung is executed, and this bit retains its state when the rung is not executed or when power cycle occurs.	**1–10**
OSR	One-Shot Rising	Triggers a one-time event.	**1–11**

continued on next page

Timer/Counter Instructions

Instruction		Purpose	Page
Mnemonic	Name		
TON	Timer On-Delay	Counts timebase intervals when the instruction is true.	**1–17**
TOF	Timer Off-Delay	Counts timebase intervals when the instruction is false.	**1–18**
RTO	Retentive Timer	Counts timebase intervals when the instruction is true and retains the accumulated value when the instruction goes false or when power cycle occurs.	**1–19**
CTU	Count Up	Increments the accumulated value at each false–to–true transition and retains the accumulated value when the instruction goes false or when power cycle occurs.	**1–24**
CTD	Count Down	Decrements the accumulated value at each false–to–true transition and retains the accumulated value when the instruction goes false or when power cycle occurs.	**1–25**
HSC	High-Speed Counter	Counts high-speed pulses from a fixed controller high-speed input.	**1–26**
RES	Reset	Resets the accumulated value and status bits of a timer or counter. Do not use with TOF timers.	**1–31**

About the Basic Instructions

These instructions, when used in ladder programs, represent hardwired logic circuits used for the control of a machine or equipment.

The basic instructions are separated into three groups: bit, timer, and counter. Before you learn about the instructions in each of these groups, we suggest that you read the overview that precedes the group:

- Bit Instructions Overview
- Timer Instructions Overview
- Counter Instructions Overview

Bit Instructions Overview

These instructions operate on a single bit of data. During operation, the processor may set or reset the bit, based on logical continuity of ladder rungs. You can address a bit as many times as your program requires.

Note *Using the same address with multiple output instructions is not recommended.*

Bit instructions are used with the following data files:

Output and Input Data Files (Files O0: and I1:)

These represent external outputs and inputs. Bits in file 1 are used to represent external inputs. In most cases, a single 16-bit word in these files will correspond to a slot location in your controller, with bit numbers corresponding to input or output terminal numbers. Unused bits of the word are not available for use.

The table below explains the addressing format for outputs and inputs. Note that the format specifies e as the slot number and s as the word number. When you are dealing with file instructions, refer to the element as e . s (slot and word), taken together.

Format	Explanation		
O:e.s/b I:e.s/b	**O**	Output	
	I	Input	
	:	Element delimiter	
	e	Slot number (decimal)	Slot 0, adjacent to the power supply in the first chassis, applies to the processor module (CPU). Succeeding slots are I/O slots, numbered from 1 to a maximum of 30.
	.	Word delimiter. Required only if a word number is necessary as noted below.	
	s	Word number	Required if the number of inputs or outputs exceeds 16 for the slot. Range: 0–255 (range accommodates multi-word “specialty cards”)
	/	Bit delimiter	
	b	Terminal number	Inputs: 0–15 Outputs: 0–15

Examples (applicable to the controller shown on page F–12):

O:3/15	Output 15, slot 3
O:5/0	Output 0, slot 5
O:10/11	Output 11, slot 10
I:7/8	Input 8, slot 7
I:2.1/3	Input 3, slot 2, word 1

Word addresses:

O:5	Output word 0, slot 5
O:5.1	Output word 1, slot 5
I:8	Input word 0, slot 8

Default Values: Your programming device will display an address more formally. For example, when you assign the address O:5/0, the programming device will show it as O:5.0/0 (Output file, slot 5, word 0, terminal 0).

Status File (File S2:)

You cannot add to or delete from the status file. The MicroLogix 1000 controller status file is explained in appendix A and the SLC 500 processor status file is explained in appendix B. You can address various bits and words as follows:

Format			
S:e/b	**S**	Status file	
	:	Element delimiter	
	e	Element number	Ranges from 0–15 in a fixed or SLC 5/01 controller, 0–32 in an SLC 5/02, 0–83 in an SLC 5/03 OS300, 0–96 in an SLC 5/03 OS301 and later and 5/04 OS400, and 0–64 in an SLC 5/04 OS401 and SLC 5/05 processors. These are 1-word elements. 16 bits per element.
	/	Bit delimiter	
	b	Bit number	Bit location within the element. Ranges from 0–15.

Examples:

S:1/15 Element 1, bit 15. This is the “first pass” bit, which you can use to initialize instructions in your program.

S:3 Element 3. The lower byte of this element is the current scan time. The upper byte is the watchdog scan time.

Bit Data File (B3:)

File 3 is the bit file, used primarily for bit (relay logic) instructions, shift registers, and sequencers. The maximum size of the file is 256 1-word elements, a total of 4096 bits. You can address bits by specifying the element number (0 to 255) and the bit number (0 to 15) within the element. You can also address bits by numbering them in sequence, 0 to 4095.

You can also address elements of this file.

Format	Explanation			Examples
Bf:e/b	**B**	Bit type file		
	f	File number. Number 3 is the default file. A file number between 9–255 can be used if additional storage is required.		**B3:3/14** Bit 14, element 3
	:	Element delimiter		
	e	Element number	Ranges from 0–255. These are 1-word elements. 16 bits per element.	**B3:252/00** Bit 0, element 252
	/	Bit delimiter		
	b	Bit number	Bit location within the element. Ranges from 0–15.	**B3:9** Bits 0–15, element 9
Format	**Explanation**			**Examples**
Bf/b	**B** **f** **/**	Same as above. Same as above. Same as above.		**B3/62** Bit 62
	b	Bit number	Numerical position of the bit within the file. Ranges from 0–4095.	**B3/4032** Bit 4032

Timer and Counter Data Files (T4: and C5:)

See pages 1–15 and 1–22 respectively for the addressing formats.

Control Data File (R6:)

These instructions use various control bits. These are 3-word elements, used with Bit Shift, FIFO, LIFO, Sequencer instructions, and ASCII instructions ABL, ACB, AHL, ARD, ARL, AWA, and AWT. Word 0 is the status word, word 1 indicates the length of stored data, and word 2 indicates position. This is shown in the following figure.

In the control element there are eight status bits and an error code byte. A fixed controller and an SLC 5/01 control element has six bits. Bits EU and EM are not used by the processor.

Control Element

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Word
EN	EU	DN	EM	ER	UL	IN	FD	Error Code								0
Length of Bit array or File (LEN)																1
Bit Pointer or Position (POS)																2

Addressable Bits

EN = Enable
EU = Unload Enable
DN = Done
EM = Stack Empty
ER = Error
UL = Unload (Bit shift only)
IN = Inhibit (This is the Running Bit [RN bit 9] for ASCII instructions)
FD = Found (SQC only)

Addressable Words

LEN = Length
POS = Position

Error Code value is displayed in HEX and is not addressable.

Assign control addresses as follows:

Format	Explanation		
Rf:e	**R**	Control file	
	f	File number. Number 6 is the default file. A file number between 9–255 can be used if additional storage is required.	
	:	Element delimiter	
	e	Element number	Ranges from 0–255. These are 3-word elements. See figure above.

Example: R6:2 Element 2, control file 6.

Address bits and words by using the format Rf:e.s/b where Rf:e is explained above, and:

. is the word delimiter
s indicates subelement
/ is the bit delimiter
b indicates bit

R6:2/15	or	**R6:2/EN**	Enable bit
R6:2/14	or	**R6:2/EU**	Unload Enable bit
R6:2/13	or	**R6:2/DN**	Done bit
R6:2/12	or	**R6:2/EM**	Stack Empty bit
R6:2/11	or	**R6:2/ER**	Error bit
R6:2/10	or	**R6:2/UL**	Unload bit
R6:2/9	or	**R6:2/IN**	Inhibit bit
R6:2/8	or	**R6:2/FD**	Found bit
R6:2.1	or	**R6:2.LEN**	Length value
R6:2.2	or	**R6:2.POS**	Position value
R6:2.1/0			Bit 0 of length value
R6:2.2/0			Bit 0 of position value

Integer Data File (N7:)

Use these addresses (at the bit level) as your program requires. These are 1-word elements, addressable at the element and bit level.

Assign integer addresses as follows:

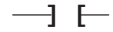
Format	Explanation		
Nf:e/b	**N**	Integer file	
	f	File number. Number 7 is the default file. A file number between 9–255 can be used if additional storage is required.	
	:	Element delimiter	
	e	Element number	Ranges from 0–255. These are 1-word elements. 16 bits per element.
	/	Bit delimiter	
	b	Bit number	Bit location within the element. Ranges from 0–15.

Examples:

N7:2	Element 2, integer file 7
N7:2/8	Bit 8 in element 2, integer file 7
N10:36	Element 36, integer file 10 (file 10 designated as an integer file by the user)

Examine if Closed (XIC)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Input Instruction

Use the XIC instruction in your ladder program to determine if a bit is On. When the instruction is executed, if the bit addressed is on (1), then the instruction is evaluated as true. When the instruction is executed, if the bit addressed is off (0), then the instruction is evaluated as false.

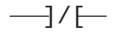
Bit Address State	XIC Instruction
0	False
1	True

Examples of devices that turn on or off include:

- a push button wired to an input (addressed as I:0/4)
- an output wired to a pilot light (addressed as O:0/2)
- a timer controlling a light (addressed as T4:3/DN)

Examine if Open (XIO)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Input Instruction

Use the XIO instruction in your ladder program to determine if a bit is Off. When the instruction is executed, if the bit addressed is off (0), then the instruction is evaluated as true. When the instruction is executed, if the bit addressed is on (1), then the instruction is evaluated as false.

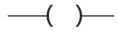
Bit Address State	XIO Instruction
0	True
1	False

Examples of devices that turn on or off include:

- motor overload normally closed (N.C.) wired to an input (I:0/10)
- an output wired to a pilot light (addressed as O:0/4)
- a timer controlling a light (addressed as T4:3/DN)

Output Energize (OTE)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

Use the OTE instruction in your ladder program to turn on a bit when rung conditions are evaluated as true.

An example of a device that turns on or off is an output wired to a pilot light (addressed as O:0/4).

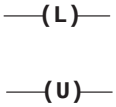
OTE instructions are reset when:

- You enter or return to the REM Run or REM Test mode or power is restored.
- The OTE is programmed within an inactive or false Master Control Reset (MCR) zone.

Note *A bit that is set within a subroutine using an OTE instruction remains set until the subroutine is scanned again.*

Output Latch (OTL) and Output Unlatch (OTU)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instructions

OTL and OTU are retentive output instructions. OTL can only turn on a bit, while OTU can only turn off a bit. These instructions are usually used in pairs, with both instructions addressing the same bit.

Your program can examine a bit controlled by OTL and OTU instructions as often as necessary.

Under fatal error conditions, physical outputs are turned off. Once the error conditions are cleared, the controller resumes operation using the data table value of the operand.

Using OTL

When you assign an address to the OTL instruction that corresponds to the address of a physical output, the output device wired to this screw terminal is energized when the bit is set (turned on or enabled).

When rung conditions become false (after being true), the bit remains set and the corresponding output device remains energized.

When enabled, the latch instruction tells the controller to turn on the addressed bit. Thereafter, the bit remains on, regardless of the rung condition, until the bit is turned off (typically by a OTU instruction in another rung).

Using OTU

When you assign an address to the OTU instruction that corresponds to the address of a physical output, the output device wired to this screw terminal is de–energized when the bit is cleared (turned off or disabled).

The unlatch instruction tells the controller to turn off the addressed bit. Thereafter, the bit remains off, regardless of the rung condition, until it is turned on (typically by a OTL instruction in another rung).

One–Shot Rising (OSR)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓

Input Instruction

The OSR instruction is a retentive input instruction that triggers an event to occur one time. Use the OSR instruction when an event must start based on the change of state of the rung from false-to-true.

When the rung conditions preceding the OSR instruction go from false-to-true, the OSR instruction will be true for one scan. After one scan is complete, the OSR instruction becomes false, even if the rung conditions preceding it remain true. The OSR instruction will only become true again if the rung conditions preceding it transition from false-to-true.

The SLC 500 and SLC 5/01 processors allow you to use one OSR instruction per output in a rung; the OSR cannot be within a branch. The SLC 5/02 and higher processors and MicroLogix 1000 controllers allow you to use one OSR instruction per output in a rung; putting the OSR within a branch is permitted.

Entering Parameters

The address assigned to the OSR instruction is *not* the one-shot address referenced by your program, nor does it indicate the state of the OSR instruction. This address allows the OSR instruction to *remember* its previous rung state.

Use a bit address from either the bit or integer data file. The addressed bit is set (1) for one scan when rung conditions preceding the OSR instruction are true (even if the OSR instruction becomes false); the bit is reset (0) when rung conditions preceding the OSR instruction are false.

Note

The bit address you use for this instruction must be unique. Do not *use it elsewhere in the program.*

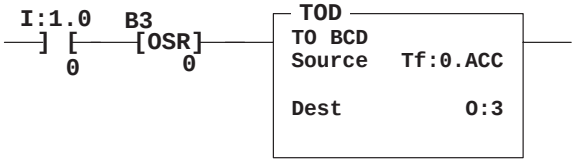
Do not *use an input or output address to program the address parameter of the OSR instruction.*

Examples

The following rungs illustrate the use of the OSR instruction. The first four rungs apply to SLC 500 and SLC 5/01 processors. The fifth rung involves output branching and applies to the SLC 5/02 and higher processors and MicroLogix 1000 controllers.

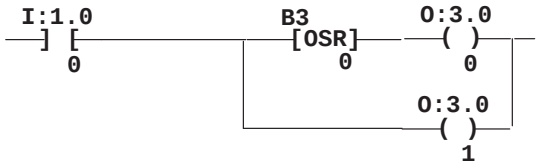
SLC 500 and SLC 5/01 Processors

When the input instruction goes from false-to-true, the OSR instruction conditions the rung so that the output goes true for one program scan. The output goes false and remains false for successive scans until the input makes another false-to-true transition.

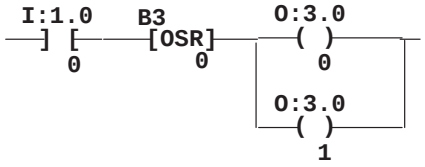


In this case, the accumulated value of a timer is converted to BCD and moved to an output word where an LED display is connected. When the timer is running, the accumulated value is changing rapidly. This value can be frozen and displayed for each false-to-true transition of the input condition of the rung.

Using an OSR Instruction in a Branch (SLC 500 and SLC 5/01 Processors)



In the above rung, the OSR instruction is not permitted inside a branch.

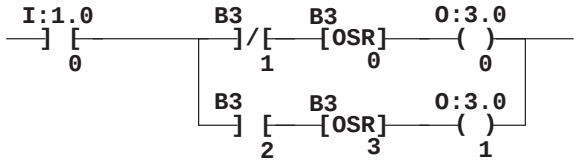


In this case, the OSR instruction is not in the branch so the rung is legal.

The SLC 500 and SLC 5/01 processors allow you to use one OSR instruction per rung.

When using a SLC 500 or SLC 5/01 processor, do not place input conditions after the OSR instruction in a rung. Unexpected operation may occur.

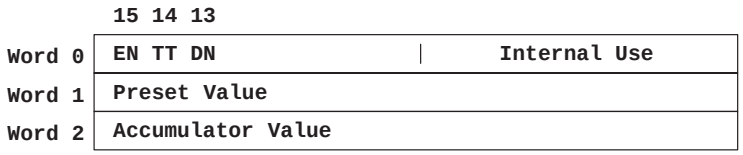
SLC 5/02 (and higher) Processors and MicroLogix 1000 Controllers



The SLC 5/02 and higher processors and MicroLogix 1000 controllers allow you to use one OSR instruction per output in a rung.

Timer Instructions Overview

Each timer address is made of a 3-word element. Word 0 is the control word, word 1 stores the preset value, and word 2 stores the accumulated value.



Addressable Bits

EN = Bit 15 Enable
TT = Bit 14 Timer Timing
DN = Bit 13 Done

Addressable Words

PRE = Preset Value
ACC = Accumulated Value

Bits labeled “Internal Use” are not addressable.

Entering Parameters

Accumulator Value (.ACC)

This is the time elapsed since the timer was last reset. When enabled, the timer updates this continually.

Preset Value (.PRE)

This specifies the value which the timer must reach before the controller sets the done bit. When the accumulated value becomes equal to or greater than the preset value, the done bit is set. You can use this bit to control an output device.

Preset and accumulated values for timers range from 0 to +32,767. If a timer preset or accumulated value is a negative number, a runtime error occurs.

Timebase

The timebase determines the duration of each timebase interval. For Fixed and SLC 5/01 processors, the timebase is set at 0.01 second. For SLC 5/02 and higher processors and MicroLogix 1000 controllers, the timebase is selectable as 0.01 (10 ms) second or 1.0 second.

Timer Accuracy

Timer accuracy refers to the length of time between the moment a timer instruction is enabled and the moment the timed interval is complete. Inaccuracy caused by the program scan can be greater than the timer timebase. You must also consider the time required to energize the output device.

Timing accuracy is –0.01 to +0 seconds, with a program scan of up to 2.5 seconds. The 1-second timer maintains accuracy with a program scan of up to 1.5 seconds. If your programs can exceed 1.5 or 2.5 seconds, repeat the timer instruction rung so that the rung is scanned within these limits.

Note *Timing could be inaccurate if Jump (JMP), Label (LBL), Jump to Subroutine (JSR), or Subroutine (SBR) instructions skip over the rung containing a timer instruction while the timer is timing. If the skip duration is less than 2.5 seconds, no time will be lost; if the skip duration exceeds 2.5 seconds, an undetectable timing error occurs. When using subroutines, a timer must be executed at least every 2.5 seconds to prevent a timing error.*

Addressing Structure

Address bits and words using the format **Tf:e.s/b**

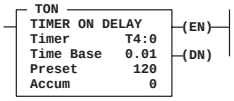
	Explanation	
T	Timer file	
f	File number. For SLC 500 processors the default is 4. A file number between 9–255 can be used for additional storage. The only valid file number is 4 for MicroLogix 1000 controllers.	
:	Element delimiter	
e	Element number	These are 3-word elements. For SLC 500 processors the range is 0–255. For MicroLogix 1000 controllers the range is from 0–39.
.	Word element	
s	Subelement	
/	Bit delimiter	
b	Bit	

Addressing Examples

- **T4:0/15** or **T4:0/EN** Enable bit
- **T4:0/14** or **T4:0/TT** Timer timing bit
- **T4:0/13** or **T4:0/DN** Done bit
- **T4:0.1** or **T4:0.PRE** Preset value of the timer
- **T4:0.2** or **T4:0.ACC** Accumulated value of the timer
- **T4:0.1/0** or **T4:0.PRE/0** Bit 0 of the preset value
- **T4:0.2/0** or **T4:0.ACC/0** Bit 0 of the accumulated value

Timer On–Delay (TON)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓



Output Instruction

Use the TON instruction to turn an output on or off after the timer has been on for a preset time interval. The TON instruction begins to count timebase intervals when rung conditions become true. As long as rung conditions remain true, the timer adjusts its accumulated value (ACC) each evaluation until it reaches the preset value (PRE). The accumulated value is reset when rung conditions go false, regardless of whether the timer has timed out.

Using Status Bits

This Bit	Is Set When	And Remains Set Until One of the Following
Timer Done Bit DN (bit 13)	accumulated value is equal to or greater than the preset value	rung conditions go false
Timer Timing Bit TT (bit 14)	rung conditions are true and the accumulated value is less than the preset value	rung conditions go false or when the done bit is set
Timer Enable Bit EN (bit 15)	rung conditions are true	rung conditions go false

When the processor changes from the REM Run or REM Test mode to the REM Program mode or user power is lost while the instruction is timing but has not reached its preset value, the following occurs:

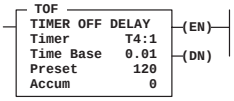
- Timer Enable (EN) bit remains set.
- Timer Timing (TT) bit remains set.
- Accumulated value (ACC) remains the same.

On returning to the REM Run or REM Test mode, the following can happen:

Condition	Result
If the rung is true:	EN bit remains set. TT bit remains set. ACC value is reset.
If the rung is false:	EN bit is reset. TT bit is reset. ACC value is reset.

Timer Off–Delay (TOF)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

Use the TOF instruction to turn an output on or off after its rung has been off for a preset time interval. The TOF instruction begins to count timebase intervals when the rung makes a true-to-false transition. As long as rung conditions remain false, the timer increments its accumulated value (ACC) each scan until it reaches the preset value (PRE). The accumulated value is reset when rung conditions go true regardless of whether the timer has timed out.

Using Status Bits

This Bit	Is Set When	And Remains Set Until One of the Following
Timer Done Bit DN (bit 13)	rung conditions are true	rung conditions go false and the accumulated value is greater than or equal to the preset value
Timer Timing Bit TT (bit 14)	rung conditions are false and the accumulated value is less than the preset value	rung conditions go true or when the done bit is reset
Timer Enable Bit EN (bit 15)	rung conditions are true	rung conditions go false

When processor operation changes from the REM Run or REM Test mode to the REM Program mode or user power is lost while a timer off-delay instruction is timing but has not reached its preset value, the following occurs:

- Timer Enable (EN) bit remains set.
- Timer Timing (TT) bit remains set.
- Timer Done (DN) bit remains set.
- Accumulated value (ACC) remains the same.

On returning to the REM Run or REM Test mode, the following can happen:

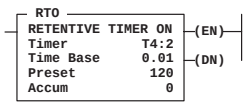
Condition	Result
If the rung is true:	TT bit is reset. DN bit remains set. EN bit is set. ACC value is reset.
If the rung is false:	TT bit is reset. DN bit is reset. EN bit is reset. ACC value is set equal to the preset value.

The Reset (RES) instruction cannot be used with the TOF instruction because RES always clears the status bits as well as the accumulated value. (See page 1–31.)

Note *The TOF times inside an inactive MCR Pair.*

Retentive Timer (RTO)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

Use the RTO instruction to turn an output on or off after its timer has been on for a preset time interval. The RTO instruction is a retentive instruction that begins to count timebase intervals when rung conditions become true.

The RTO instruction retains its accumulated value when any of the following occurs:

- Rung conditions become false.
- You change processor operation from the REM Run or REM Test mode to the REM Program mode.
- The processor loses power (provided that battery backup is maintained).
- A fault occurs.

When you return the processor to the REM Run or REM Test mode and/or rung conditions go true, timing continues from the retained accumulated value. By retaining its accumulated value, retentive timers measure the cumulative period during which rung conditions are true.

Using Status Bits

This Bit	Is Set When	And Remains Set Until One of the Following
Timer Done Bit DN (bit 13)	accumulated value is equal to or greater than the preset value	the appropriate RES instruction is enabled
Timer Timing Bit TT (bit 14)	rung conditions are true and the accumulated value is less than the preset value	rung conditions go false or when the done bit is set
Timer Enable Bit EN (bit 15)	rung conditions are true	rung conditions go false

Note

To reset the retentive timer's accumulated value and status bits after the RTO rung goes false, you must program a reset (RES) instruction with the same address in another rung.

When the processor changes from the REM Run or REM Test mode to the REM Program or REM Fault mode, or user power is lost while the timer is timing but not yet at the preset value, the following occurs:

- Timer Enable (EN) bit remains set.
- Timer Timing (TT) bit remains set.
- Accumulated value (ACC) remains the same.

On returning to the REM Run or REM Test mode or when power is restored, the following can happen:

Condition	Result
If the rung is true:	TT bit remains set. EN bit remains set. ACC value remains the same and resumes incrementing.
If the rung is false:	TT bit is reset. DN bit remains in its last state. EN bit is reset. ACC value remains in its last state.

Using Counters

Counter Data File Elements

Each Counter address is made of a 3-word data file element. Word 0 is the control word, containing the status bits of the instruction. Word 1 is the preset value. Word 2 is the accumulated value.

The control word for counter instructions includes six status bits, as indicated below.

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Word 0	CU	CD	DN	OV	UN	UA			Internal Use							
Word 1	Preset Value															
Word 2	Accumulated Value															

Addressable Bits

CU = Count up enable
CD = Count down enable
DN = Done bit
OV = Overflow bit
UN = Underflow bit
UA = Update accum. value
(HSC in fixed controller only)

Addressable Words

PRE = Preset
ACC = Accum

Bits labeled “Internal Use” are not addressable.

For information on the MicroLogix 1000 controller high-speed counter instruction, see chapter 7.

Entering Parameters

Accumulator Value (.ACC)

This is the number of false-to-true transitions that have occurred since the counter was last reset.

Preset Value (PRE)

Specifies the value which the counter must reach before the controller sets the done bit. When the accumulator value becomes equal to or greater than the preset value, the done status bit is set. You can use this bit to control an output device.

Preset and accumulated values for counters range from –32,768 to +32,767, and are stored as signed integers. Negative values are stored in two's complement form.

Addressing Structure

Assign counter addresses using the format **Cf:e.s/b**

Explanation		
C	Counter	
f	File number. For SLC 500 processors the default is 5. A file number between 9–255 can be used for additional storage. The only valid file number is 5 for MicroLogix 1000 controllers.	
:	Element delimiter	
e	Element number	These are 3-word elements. For SLC 500 processors the range is 0–255. For MicroLogix 1000 controllers the range is from 0–31.
.	Word element	
s	Subelement	
/	Bit delimiter	
b	Bit	

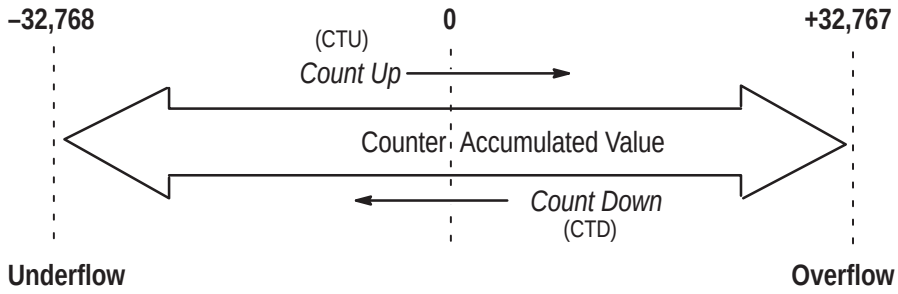
Examples

- **C5:0/15** or **C5:0/CU** Count up enable bit
- **C5:0/14** or **C5:0/CD** Count down enable bit
- **C5:0/13** or **C5:0/DN** Done bit
- **C5:0/12** or **C5:0/OV** Overflow bit
- **C5:0/11** or **C5:0/UN** Underflow bit
- **C5:0/10** or **C5:0/UA** Update accum. bit (HSC in fixed controller only)
- **C5:0.1 or C5:0PRE** Preset value of the counter
- **C5:0.2** or **C5:0.ACC** Accumulated value of the counter
- **C5:0.1/0** or **C5:0.PRE/0** Bit of the preset value
- **C5:0.2/0** or **C5:0.ACC/0** Bit 0 of the accumulated value

How Counters Work

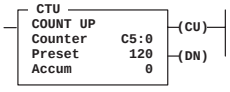
The figure below demonstrates how a counter works. The count value must remain in the range of –32768 to +32767. If the count value goes above +32767 or below –32768, a counter status overflow (OV) or underflow (UN) bit is set.

A counter can be reset to zero using the reset (RES) instruction.



Count Up (CTU)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

The CTU is an instruction that counts false-to-true rung transitions. Rung transitions can be caused by events occurring in the program (from internal logic or by external field devices) such as parts traveling past a detector or actuating a limit switch.

When rung conditions for a CTU instruction have made a false-to-true transition, the accumulated value is incremented by one count, provided that the rung containing the CTU instruction is evaluated between these transitions. The ability of the counter to detect false–to–true transitions depends on the speed (frequency) of the incoming signal.

Note *The on and off duration of an incoming signal must not be faster than the scan time 2x (assuming a 50% duty cycle).*

The accumulated value is retained when the rung conditions again become false. The accumulated count is retained until cleared by a reset (RES) instruction that has the same address as the counter reset.

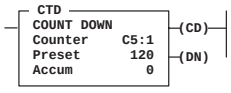
Using Status Bits

This Bit	Is Set When	And Remains Set Until One of the Following
Count Up Overflow Bit OV (bit 12)	accumulated value wraps around to –32,768 (from +32,767) and continues counting up from there	a RES instruction having the same address as the CTU instruction is executed OR the count is decremented less than or equal to +32,767 with a CTD instruction
Done Bit DN (bit 13)	accumulated value is equal to or greater than the preset value	the accumulated value becomes less than the preset value
Count Up Enable Bit CU (bit 15)	rung conditions are true	rung conditions go false OR a RES instruction having the same address as the CTU instruction is enabled

The accumulated value is retained after the CTU instruction goes false, or when power is removed from and then restored to the controller. Also, the on or off status of counter done, overflow, and underflow bits is retentive. The accumulated value and control bits are reset when the appropriate RES instruction is enabled. The CU bits are always set prior to entering the REM Run or REM Test modes.

Count Down (CTD)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

The CTD is an instruction that counts false-to-true rung transitions. Rung transitions can be caused by events occurring in the program such as parts traveling past a detector or actuating a limit switch.

When rung conditions for a CTD instruction have made a false-to-true transition, the accumulated value is decremented by one count, provided that the rung containing the CTD instruction is evaluated between these transitions.

The accumulated counts are retained when the rung conditions again become false. The accumulated count is retained until cleared by a reset (RES) instruction that has the same address as the counter reset.

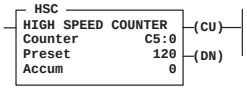
Using Status Bits

This Bit	Is Set When	And Remains Set Until One of the Following
Count Down Underflow Bit UN (bit 11)	accumulated value wraps around to +32,767 (from –32,768) and continues counting down from there	a RES instruction having the same address as the CTD instruction is enabled. OR the count is incremented greater than or equal to +32,767 with a CTU instruction
Done Bit DN (bit 13)	accumulated value is equal to or greater than the preset value	the accumulated value becomes less than the preset
Count Down Enable Bit CD (bit 14)	rung conditions are true	rung conditions go false OR a RES instruction having the same address as the CTD instruction is enabled

The accumulated value is retained after the CTD instruction goes false, or when power is removed from and then restored to the controller. Also, the on or off status of counter done, overflow, and underflow bits is retentive. The accumulated value and control bits are reset when the appropriate RES instruction is executed. The CD bits are always set prior to entering the REM Run or REM Test modes.

High-Speed Counter (HSC)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓					



Output Instruction

The High-Speed Counter is a variation of the CTU counter. The HSC instruction is enabled when the rung logic is true and disabled when the rung logic is false.

For information on using the high-speed counter instruction, see chapter 7.

Note *The HSC instruction counts transitions that occur at input terminal I:0/0. The HSC instruction does* not *count rung transitions. You enable or disable the HSC rung to enable or disable the counting of transitions occurring at input terminal I:0/0. We recommend placing the HSC instruction in an unconditional rung. Do not place the XIC instruction with address I:0/0 in series with the HSC instruction because counts will be lost.*

The HSC is a special CTU counter for use with 24 VDC SLC fixed and 24 VDC MicroLogix 1000 controllers. The HSC's status bits and accumulated values are non-retentive.

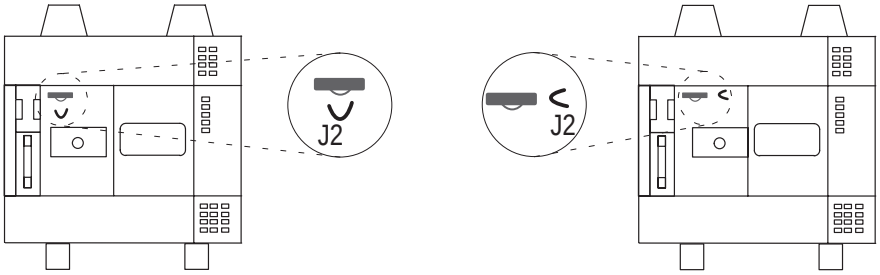
Note *This instruction provides high-speed counting for fixed I/O controllers with 24 VDC inputs. One HSC instruction is allowed per controller. To use the instruction, you must cut the jumper as shown below. A shielded cable is recommended to reduce noise to the input.*

High-Speed Counter Operation

For high-speed counter operation you must do the following:

1. Turn off power to the fixed controller.
2. Remove the SLC 500 cover.
3. Locate and cut jumper wire J2. Do not remove completely but make certain that the ends of the cut jumper wire are not touching each other.

The High–Speed Counter jumper is located either beneath
the battery connector OR to the right of the battery connector.



4. Replace the cover.

Note

Input I:0/0 then operates in the high-speed mode. The address of the high–speed counter enable bit is C5:0/CU. When rung conditions are true, C5:0/CU is set and transitions occurring at input I:0/0 are counted.

To begin high-speed counting, load a preset value into C5:0.PRE and enable the counter rung. To load a preset value, do one of the following:

- Change to the REM Run or REM Test mode from another mode.
- Power up the processor in the REM Run mode.
- Reset the HSC using the RES instruction.

Automatic reloading occurs when the HSC itself sets the DN bit on interrupt.

Each input transition that occurs at input I:0/0 causes the HSC accumulated value to increment. When the accumulated value equals the preset value, the Done bit (C5:0/DN) is set, the accumulated value is cleared, and the preset value (C5:0.PRE) is loaded into the HSC in preparation for the next high-speed transition at input I:0/0.

Your ladder program should poll the Done bit (C5:0/DN) to determine the state of the HSC. Once the Done bit has been detected as set, the ladder program should clear bit C5:0/DN (using the unlatch OTU instruction) before the HSC accumulated again reaches the preset value, or the overflow bit (C5:0/OV) will be set.

The HSC differs from the CTU and CTD counters. The CTU and CTD are software counters. The HSC is a hardware counter and operates asynchronously to the ladder program scan. The HSC accumulated value (C5:0.ACC) is normally updated each time the HSC rung is evaluated in the ladder program. This means that the HSC hardware accumulator value is transferred to the HSC software accumulator. Only use the OTE instruction to transfer this value. The HSC instruction immediately clears bit C5:0/UA following the accumulated update.

Many HSC counts may occur between HSC evaluations, which would make C5:0.ACC inaccurate when used throughout a ladder program. To allow for an accurate HSC accumulated value, the update accumulator bit (C5:0/UA) causes C5:0.ACC to be immediately updated to the state of the hardware accumulator when set.

Use the RES instruction to reset the high–speed counter at address C5:0. The HSC instruction clears the status bits, the accumulator, and loads the preset value during:

- power up
- entry into the REM Run mode
- a reset

High-Speed Counter Data Elements

Address C5:0 is the HSC counter 3-word element.

	15 14 13 12 11 10 09 08	07 06 05 04 03 02 01 00
Word 0	CU CD DN OV UN UA	Not Used
Word 1	Preset Value	
Word 2	Accumulator Value	

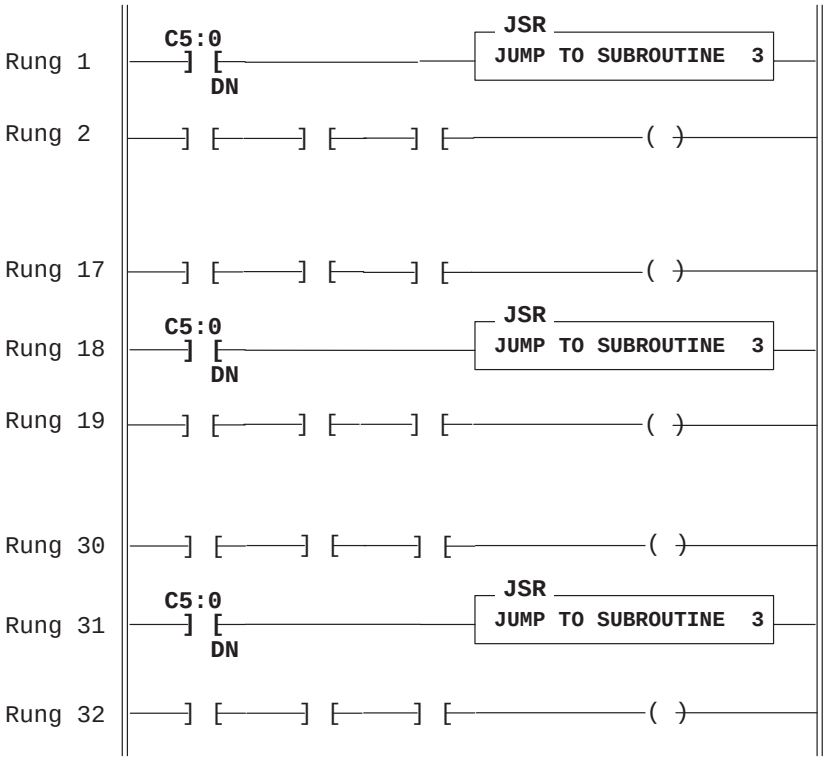
CU = Counter up enable bit
CD = Counter down enable bit
DN = Done bit
OV = Overflow bit
UN = Underflow bit
UA = Update accumulator (HSC only)

- Word 0 contains the following status bits of the HSC instruction:
  - Bit 10 (UA) updates the accumulator word of the HSC to reflect the immediate state of the HSC when true.
  - Bit 12 (OV) indicates if a HSC overflow has occurred.
  - Bit 13 (DN) indicates if the HSC preset value has been reached.
  - Bit 15 (CU) shows the Enable/Disable state of the HSC instruction.
- Word 1 contains the preset value that is loaded into the HSC when either the RES instruction is executed, when the Done bit is set, or when powerup takes place. The valid range is +1 to +32767.
- Word 2 contains the HSC accumulator value. This word is updated each time the HSC instruction is evaluated and when the update accumulator bit is set using an OTE instruction. This accumulator is read only. Any value written to the accumulator is overwritten by the actual high–speed counter on instruction evaluation, reset, or REM Run mode entry.

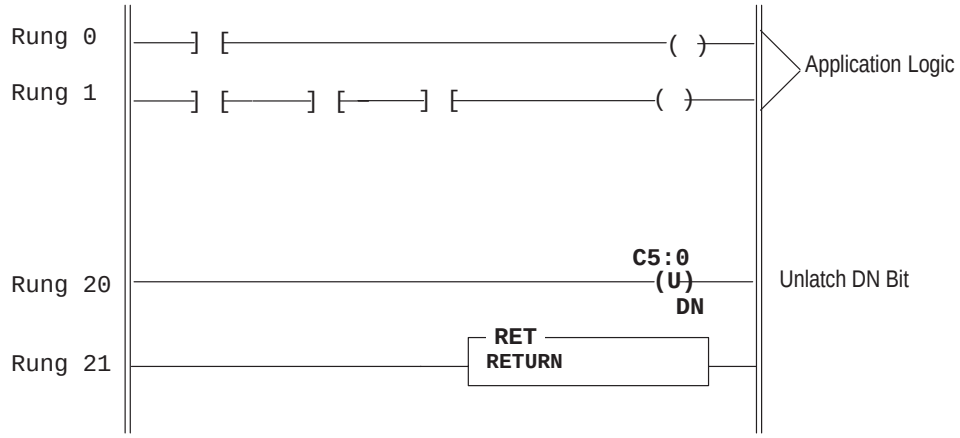
Application Example

In the following figures, rungs 1, 18, and 31 of the main program file each consist of an XIC instruction addressed to the HSC done bit and a JSR instruction. These rungs poll the status of the HSC done bit. When the Done bit is set at any of these poll points, program execution moves to subroutine file 3, executing the HSC logic. After the HSC logic is executed, the Done bit is reset by an unlatch instruction, and program execution returns to the main program file.

Application Example – File 2 (Poll for DN Bit in Main Program)



Application Example – File 3 (Execute HSC Logic)



Reset (RES)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓

—(RES)—
Output Instruction

Use a RES instruction to reset a timer or counter. When the RES instruction is enabled, it resets the Timer On Delay (TON), Retentive Timer (RTO), Count Up (CTU), or Count Down (CTD) instruction having the same address as the RES instruction.

Using a RES instruction for a:	The processor resets the:
Timer (Do not use a RES instruction with a TOF.)	ACC value to 0 DN bit TT bit EN bit
Counter	ACC value to 0 OV bit UN bit DN bit CU bit CD bit
Control	POS value to 0 EN bit EU bit DN bit EM bit ER bit UL bit IN and FD go to last state

Note

If using this instruction to reset the MicroLogix 1000 controller HSC accumulator, see page 7–22.

When resetting a counter, if the RES instruction is enabled and the counter rung is enabled, the CU or CD bit is reset.

If the counter preset value is negative, the RES instruction sets the accumulated value to zero. This in turn causes the done bit to be set by a count down or count up instruction.

Because the RES instruction resets the accumulated value, and the done, timing, and enabled bits, do *not* use the RES instruction to reset a timer address used in a TOF instruction. Otherwise, unpredictable machine operation or injury to personnel may occur.

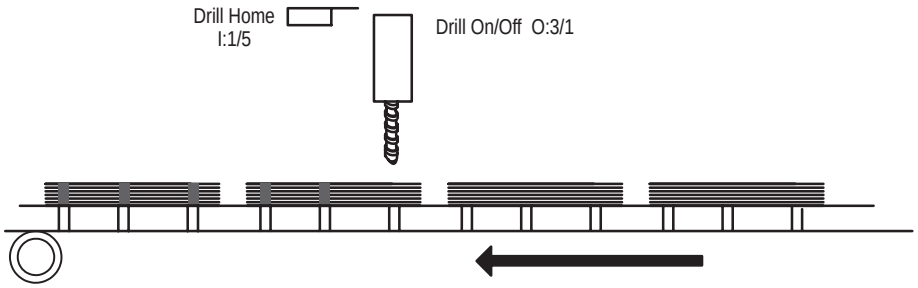
Basic Instructions in the Paper Drilling Machine Application Example

This section provides ladder rungs to demonstrate the use of basic instructions. The rungs are part of the paper drilling machine application example described in appendix H. You will be adding the main program in file 2 and adding a subroutine to file 6.

Adding File 2

The rungs shown on the following page are referred to as the program's "start–up" logic. They determine the conditions necessary to start the machine in motion by monitoring the start and stop push buttons. When the start push button is pressed, it enables the conveyor to move and starts spinning the drill bit. When the stop push button is pressed, it disables the conveyor motion and turns off the drill motor.

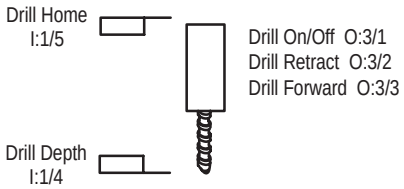
The start–up logic also checks to make sure that the drill is fully retracted (in the home position) and that the drill bit is not past its useful life (determined elsewhere in the program) before allowing the conveyor to move.



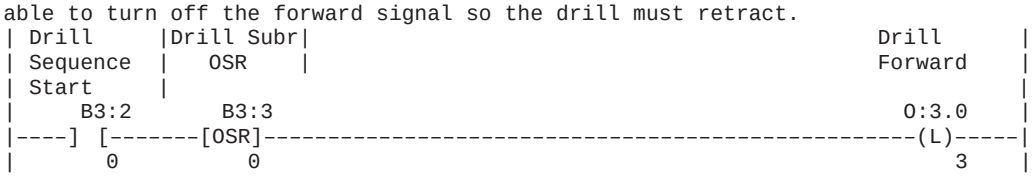
```
Rung 2:0
These rungs will start the conveyer in motion when the start button is
pressed.  However, there are other conditions that must also be met before we
start the conveyor.  They are:  the drill must be in its fully retracted
position (home) and the drill bit must not be past its maximum useful life.
These rungs will also stop the conveyor when the stop button is pressed or when
the drill life is exceeded.
|   START     |Drill        STOP                                      Machine    |
|   Button    |Home LS      Button                                      RUN      |
|                                                                     Latch      |
|     I:1.0      I:1.0        I:1.0                                      B3:0   |
|-+----] [--------] [-----+----]/[-------------------------------------( )-----|
| |       6          5    |       7                                      0    |
| | Machine               |                                                   |
| |   RUN                 |                                                   |
| | Latch                 |                                                   |
| |     B3:0              |                                                   |
| +----] [----------------+                                                   |
|         0                                                                   |
Rung 2:1
| Machine                                                          Drill      |
|   RUN                                                            Motor ON   |
| Latch                                                                       |
|     B3:0                                                           O:3.0    |
|----] [---------------------------------------------+------------( )-----+-|
|       0                                            |               1    | |
|                                                    |         Conveyor   | |
|                                                    |         Start/Stop | |
|                                                    |                    | |
|                                                    |  B3:0     O:3.0    | |
|                                                    +--[OSR]-----(L)-----+ |
|                                                          1        0      |
Rung 2:2
Stop the conveyor if any conditions exist that unlatch the ”Machine RUN Latch”
bit.
| Machine   |                                                         Conveyor   |
|   RUN     |                                                         Start/Stop |
| Latch     |                                                                    |
|     B3:0                                                              O:3.0    |
|----]/[-------------------------------------------------------------------(U)-----|
|       0                                                                  0    |
```

Adding File 6

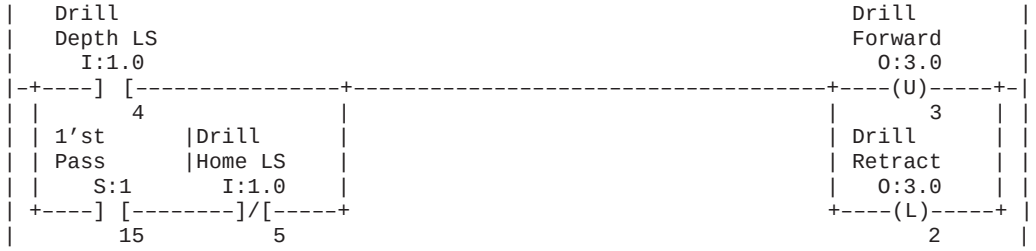
This subroutine controls the up and down motion of the drill for the paper drilling machine.



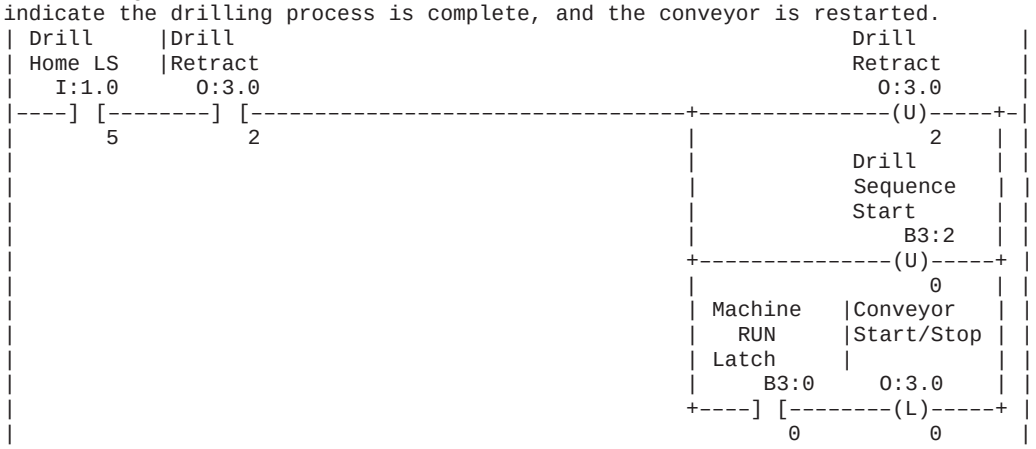
Rung 6:0
This section of ladder logic controls the up/down motion of the drill for the book drilling machine.
When the conveyor positions the book under the drill, the DRILL SEQUENCE START bit is set. This rung uses that bit to begin the drilling operation. Because the bit is set for the entire drilling operation, the OSR is required to be able to turn off the forward signal so the drill must retract.



Rung 6:1
When the drill has drilled through the book, the body of the drill will actuate the DRILL DEPTH limit switch. When this happens, the DRILL FORWARD signal is turned off and the DRILL RETRACT signal is turned on.



Rung 6:2
When the drill is retracting (after drilling a hole), the body of the drill will actuate the DRILL HOME limit switch. When this happens the DRILL RETRACT signal is turned off, the DRILL SEQUENCE START bit is turned off to indicate the drilling process is complete, and the conveyor is restarted.



2 Comparison Instructions

This chapter contains general information about comparison instructions and explains how they function in your application program. Each of the comparison instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the comparison instructions in use.

Comparison Instructions

Instruction		Purpose	Page
Mnemonic	Name		
EQU	Equal	Test whether two values are equal.	**2–3**
NEQ	Not Equal	Test whether one value is not equal to a second value.	**2–3**
LES	Less Than	Test whether one value is less than a second value.	**2–3**
LEQ	Less Than or Equal	Test whether one value is less than or equal to a second value.	**2–4**
GRT	Greater Than	Test whether one value is greater than another.	**2–4**
GEQ	Greater Than or Equal	Test whether one value is greater than or equal to a second value.	**2–4**
MEQ	Masked Comparison for Equal	Test portions of two values to see whether they are equal. Compares 16-bit data of a source address to 16-bit data at a reference address through a mask.	**2–5**
LIM	Limit Test	Test whether one value is within the limit range of two other values.	**2–5**

About the Comparison Instructions

Comparison instructions are used to test pairs of values to condition the logical continuity of a rung. As an example, suppose a LES instruction is presented with two values. If the first value is less than the second, then the comparison instruction is true.

To learn more about the compare instructions, we suggest that you read the Compare Instructions Overview that follows.

Comparison Instructions Overview

The following general information applies to comparison instructions.

Using Indexed Word Addresses

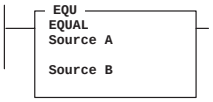
When using comparison instructions, you have the option of using indexed word addresses for instruction parameters specifying word addresses. Indexed addressing is discussed in appendix F of this manual.

Using Indirect Word Addresses

You have the option of using indirect word-level and bit-level addresses for instructions specifying word addresses when using an SLC 5/03 OS302, SLC 5/04 OS401, or SLC 5/05 processors. See appendix F for more information.

Equal (EQU)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



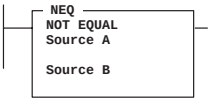
Input Instruction

Use the EQU instruction to test whether two values are equal. If source A and source B are equal, the instruction is logically true. If these values are not equal, the instruction is logically false.

Source A must be an address. Source B can either be a program constant or a address. Negative integers are stored in two's complementary form.

Not Equal (NEQ)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



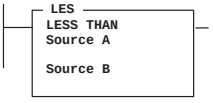
Input Instruction

Use the NEQ instruction to test whether two values are not equal. If source A and source B are not equal, the instruction is logically true. If the two values are equal, the instruction is logically false.

Source A must be an address. Source B can be either a program constant or an address. Negative integers are stored in two's complementary form.

Less Than (LES)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



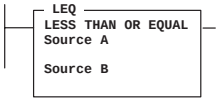
Input Instruction

Use the LES instruction to test whether one value (source A) is less than another (source B). If source A is less than the value at source B, the instruction is logically true. If the value at source A is greater than or equal to the value at source B, the instruction is logically false.

Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complementary form.

Less Than or Equal (LEQ)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



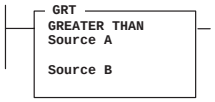
Input Instruction

Use the LEQ instruction to test whether one value (source A) is less than or equal to another (source B). If the value at source A is less than or equal to the value at source B, the instruction is logically true. If the value at source A is greater than the value at source B, the instruction is logically false.

Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complementary form.

Greater Than (GRT)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



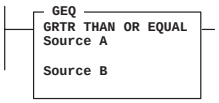
Input Instruction

Use the GRT instruction to test whether one value (source A) is greater than another (source B). If the value at source A is greater than the value at source B, the instruction is logically true. If the value at source A is less than or equal to the value at source B, the instruction is logically false.

Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complementary form.

Greater Than or Equal (GEQ)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



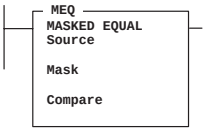
Input Instruction

Use the GEQ instruction to test whether one value (source A) is greater than or equal to another (source B). If the value at source A is greater than or equal to the value at source B, the instruction is logically true. If the value at source A is less than the value at source B, the instruction is logically false.

Source A must be an address. Source B can either be a program constant or an address. Negative integers are stored in two's complementary form.

Masked Comparison for Equal (MEQ)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Input Instruction

Use the MEQ instruction to compare data at a source address with data at a compare address. Use of this instruction allows portions of the data to be masked by a separate word.

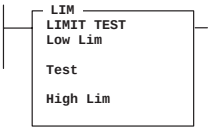
Entering Parameters

- **Source** is the address of the value you want to compare.
- **Mask** is the address of the mask through which the instruction moves data. The mask can be a hexadecimal value.
- **Compare** is an integer value or the address of the reference.

If the 16 bits of data at the source address are equal to the 16 bits of data at the compare address (less masked bits), the instruction is true. The instruction becomes false as soon as it detects a mismatch. Bits in the mask word mask data when reset; they pass data when set.

Limit Test (LIM)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓



Input Instruction

Use the LIM instruction to test for values within or outside a specified range, depending on how you set the limits.

Entering Parameters

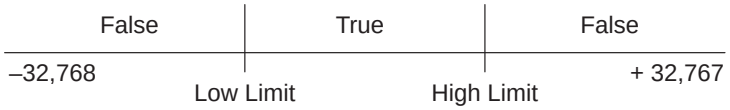
The Low Limit, Test, and High Limit values can be word addresses or constants, restricted to the following combinations:

- If the Test parameter is a program constant, both the Low Limit and High Limit parameters must be word addresses.

- If the Test parameter is a word address, the Low Limit and High Limit parameters can be either a program constant or a word address.

True/False Status of the Instruction

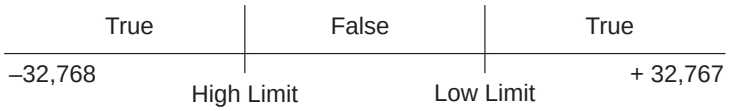
If the Low Limit has a value equal to or less than the High Limit, the instruction is true when the Test value is between the limits or is equal to either limit. If the Test value is outside the limits, the instruction is false, as shown below.



Example, low limit less than high limit:

Low Limit	High Limit	Instruction is True when Test value is	Instruction is False when Test value is
5	8	5 through 8	–32,768 through 4 and 9 through 32,767

If the Low Limit has a value greater than the High Limit, the instruction is false when the Test value is between the limits. If the Test value is equal to either limit or outside the limits, the instruction is true, as shown below.



Example, low limit greater than high limit:

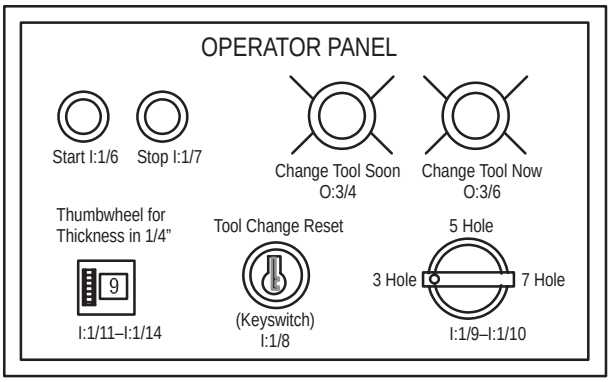
Low Limit	High Limit	Instruction is True when Test value is	Instruction is False when Test value is
8	5	–32,768 through 5 and 8 through 32,767	6 and 7

Comparison Instructions in the Paper Drilling Machine Application Example

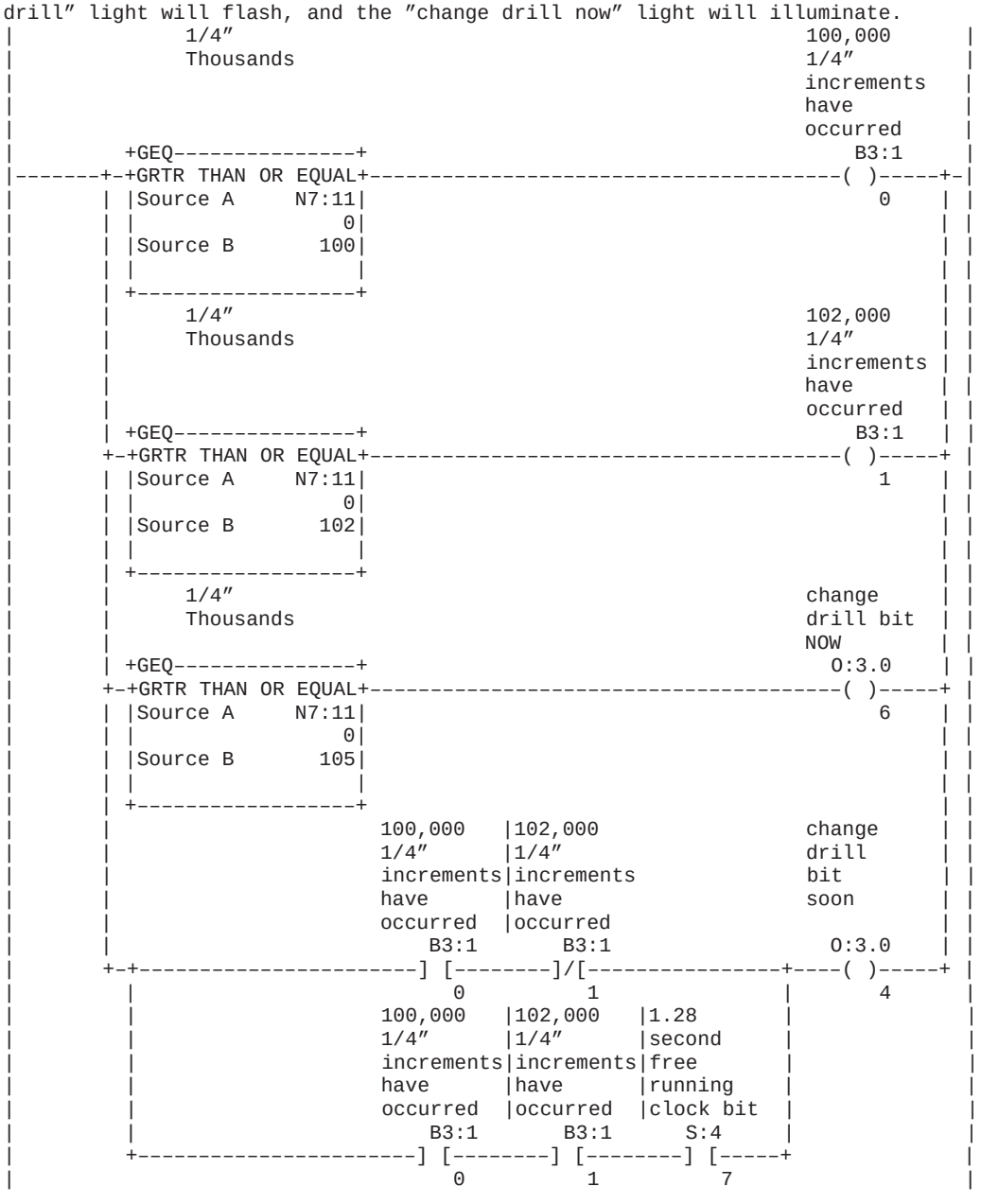
This section provides ladder rungs to demonstrate the use of comparison instructions. The rungs are part of the paper drilling machine application example described in appendix H.

Beginning a Subroutine in File 7

This section of ladder keeps track of the total inches of paper the current drill bit has drilled through. As the current bit wears out, a light illuminates on the operator panel, as shown below, to warn the operator to change the drill bit.



Rung 7:0
This rung examines the number of 1/4″ thousands that have accumulated over the life of the current drill bit. If the bit has drilled between 100,000–101,999 1/4″ increments of paper, then the ″change drill″ light will illuminate steady. When the value is between 102,000–103,999, then the ″change drill″ light will flash at a 1.28 second rate. When the value reaches 105,000, then the ″change drill″ light will flash, and the ″change drill now″ light will illuminate.



3 Math Instructions

This chapter contains general information about math instructions and explains how they function in your logic program. Each of the math instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the math instructions in use.

Math Instructions

Instruction		Purpose	Page
Mnemonic	**Name**		
ADD	Add	Adds source A to source B and stores the result in the destination.	**3–6**
SUB	Subtract	Subtracts source B from source A and stores the result in the destination.	**3–6**
MUL	Multiply	Multiplies source A by source B and stores the result in the destination.	**3–10**
DIV	Divide	Divides source A by source B and stores the result in the destination and the math register.	**3–11**
DDV	Double Divide	Divides the contents of the math register by the source and stores the result in the destination and the math register.	**3–12**
CLR	Clear	Sets all bits of a word to zero.	**3–13**
SQR	Square Root	Calculates the square root of the source and places the integer result in the destination.	**3–13**
SCP	Scale with Parameters	Produces a scaled output value that has a linear relationship between the input and scaled values.	**3–14**

continued on next page

Instruction		Purpose	Page
Mnemonic	Name		
SCL	Scale Data	Multiplies the source by a specified rate, adds to an offset value, and stores the result in the destination.	**3–17**
ABS	Absolute	Calculates the absolute value of the source and places the result in the destination.	**3–24**
CPT	Compute	Evaluates an expression and stores the result in the destination.	**3–25**
SWP	Swap	Swaps the low and high bytes of a specified number of words in a bit, integer, ASCII, or string file.	**3–27**
ASN	Arc Sine	Takes the arc sine of a number and stores the result (in radians) in the destination.	**3–28**
ACS	Arc Cosine	Takes the arc cosine of a number and stores the result (in radians) in the destination.	**3–28**
ATN	Arc Tangent	Takes the arc tangent of a number and stores the result (in radians) in the destination.	**3–29**
COS	Cosine	Takes the cosine of a number and stores the result in the destination.	**3–29**
LN	Natural Log	Takes the natural log of the value in the source and stores it in the destination.	**3–30**
LOG	Log to the Base 10	Takes the log base 10 of the value in the source and stores the result in the destination.	**3–30**
SIN	Sine	Takes the sine of a number and stores the result in the destination.	**3–31**
TAN	Tangent	Takes the tangent of a number and stores the result in the destination.	**3–31**
XPY	X to the Power of Y	Raise a value to a power and stores the result in the destination.	**3–32**

About the Math Instructions

The majority of the instructions take two input values, perform the specified arithmetic function, and output the result to an assigned memory location.

For example, both the ADD and SUB instructions take a pair of input values, add or subtract them, and place the result in the specified destination. If the result of the operation exceeds the allowable value, an overflow or underflow bit is set.

To learn more about the math instructions, we suggest that you read the Math Instructions Overview that follows.

Math Instructions Overview

The following general information applies to math instructions.

Entering Parameters

- **Source** is the address(es) of the value(s) on which the mathematical, logical, or move operation is to be performed. This can be word addresses or program constants. An instruction that has two source operands does not accept program constants in both operands.
- **Destination** is the address of the result of the operation. Signed integers are stored in two's complementary form and apply to both source and destination parameters.

  When using either an SLC 5/03 (OS301 and higher), SLC 5/04, or SLC 5/05 processor; floating point and string values (specified at the word level) are supported. Refer to appendix E for additional valid addressing types.

Using Indexed Word Addresses

You have the option of using indexed word addresses for instruction parameters specifying word addresses (except for fixed and SLC 5/01 processors). Indexed addressing is discussed in appendix F.

Using Indirect Word Addresses

You have the option of using indirect word-level and bit-level addresses for instructions specifying word addresses when using an SLC 5/03 (OS302), SLC 5/04 (OS401), or SLC 5/05 processors. See appendix C for more information.

Updates to Arithmetic Status Bits

The arithmetic status bits are found in Word 0, bits 0–3 in the controller status file. After an instruction is executed, the arithmetic status bits in the status file are updated:

With this Bit:		The Controller:
S:0/0	Carry (C)	sets if carry is generated; otherwise cleared.
S:0/1	Overflow (V)	indicates that the actual result of a math instruction does not fit in the designated destination.
S:0/2	Zero (Z)	indicates a 0 value after a math, move, or logic instruction.
S:0/3	Sign (S)	indicates a negative (less than 0) value after a math, move, or logic instruction.

Overflow Trap Bit, S:5/0

Minor error bit (S:5/0) is set upon detection of a mathematical overflow or division by zero. If this bit is set upon execution of an END statement or a Temporary End (TND) instruction, or an I/O Refresh (REF), the recoverable major error code 0020 is declared.

In applications where a math overflow or divide by zero occurs, you can avoid a CPU fault by using an unlatch (OTU) instruction with address S:5/0 in your program. The rung must be between the overflow point and the END, TND, or REF statement.

Changes to the Math Register, S:13 and S:14

Status word S:13 contains the *least* significant word of the 32-bit values of the MUL and DDV instructions. It contains the remainder for DIV and DDV instructions. It also contains the first four BCD digits for the Convert from BCD (FRD) and Convert to BCD (TOD) instructions.

Status word S:14 contains the *most* significant word of the 32-bit values of the MUL and DDV instructions. It contains the unrounded quotient for DIV and DDV instructions. It also contains the most significant digit (digit 5) for TOD and FRD instructions.

Using Floating Point Data File (F8:)

This file type is valid for SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors. These are 2-word elements and addressable only at the element level.

Assign floating point addresses as follows:

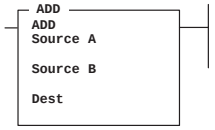
Format	Explanation		
Ff:e	**F**	Floating Point file	
	f	File number. Number 8 is the default file. A file number between 9- 255 can be used if additional storage is required.	
	:	Element delimiter	
	e	Element number	Ranges from 0- 255. These are 2-word elements. Non-extended 32-bit numbers

Examples:

F8:2	Element 2, floating point file 8
F10:36	Element 36, floating point file 10 (file 10 designated as a floating point file by the user)

Add (ADD)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

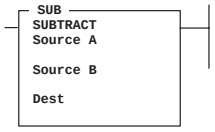
Use the ADD instruction to add one value (source A) to another value (source B) and place the result in the destination.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	sets if carry is generated; otherwise resets (integer). For floating point, it is cleared.
Overflow (V)	sets if overflow is detected at destination; otherwise resets. On overflow, the minor error flag is also set. For floating point, the overflow value is placed in the destination. For an integer, the value –32,768 or 32,767 is placed in the destination. Exception: If you are using an SLC 5/02 or higher processor or a MicroLogix 1000 controller and have S:2/14 (math overflow selection bit) set, then the unsigned, truncated overflow remains in the destination.
Zero (Z)	sets if result is zero; otherwise resets.
Sign (S)	sets if result is negative; otherwise resets.

Subtract (SUB)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



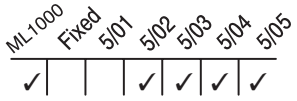
Output Instruction

Use the SUB instruction to subtract one value (source B) from another (source A) and place the result in the destination.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	sets if borrow is generated; otherwise resets (integer). For floating point it is cleared.
Overflow (V)	sets if underflow; otherwise reset. On underflow, the minor error flag is also set. For floating point, the overflow value is placed in the destination. For an integer, the value –32,768 or 32,767 is placed in the destination. Exception: If you are using an SLC 5/02 or higher processor or a MicroLogix 1000 controller and have S:2/14 (math overflow selection bit) set, then the unsigned, truncated overflow remains in the destination.
Zero (Z)	sets if result is zero; otherwise resets.
Sign (S)	sets if result is negative; otherwise resets.

32-Bit Addition and Subtraction



You have the option of performing 16-bit or 32-bit signed integer addition and subtraction. This is facilitated by status file bit S:2/14 (math overflow selection bit).

Math Overflow Selection Bit S:2/14

Set this bit when you intend to use 32-bit addition and subtraction. When S:2/14 is set, and the result of an ADD, SUB, MUL, DIV, or NEG instruction cannot be represented in the destination address (due to math underflow or overflow):

- The overflow bit S:0/1 is set.
- The overflow trap bit S:5/0 is set.
- The destination address contains the unsigned, truncated, least significant 16 bits of the result.

Note *For MUL, DIV, integer, and all floating point instructions with an integer destination, when S:2/14 is set, the state change takes effect immediately.*

When S:2/14 is reset (default condition), and the result of an ADD, SUB, MUL, DIV, or NEG instruction cannot be represented in the destination address (due to math underflow or overflow):

- The overflow bit S:0/1 is set.
- The overflow trap bit S:5/0 is set.
- The destination address contains 32767 if the result is positive or –32768 if the result is negative.

Note *Additionally, the SLC 5/03 and higher processors only assert the state of bit S:2/14 at the end of scan for the ADD, SUB, and NEG instructions.*

Note that the status of bit S:2/14 has no effect on the DDV instruction. Also, it has no effect on the math register content when using MUL and DIV instructions.

Note *The SLC 5/03 and higher processors only interrogate this bit upon going to the Run mode and end-of-scan. Use the Data Monitor function to make this selection prior to entering the Run mode.*

Example of 32-bit Addition

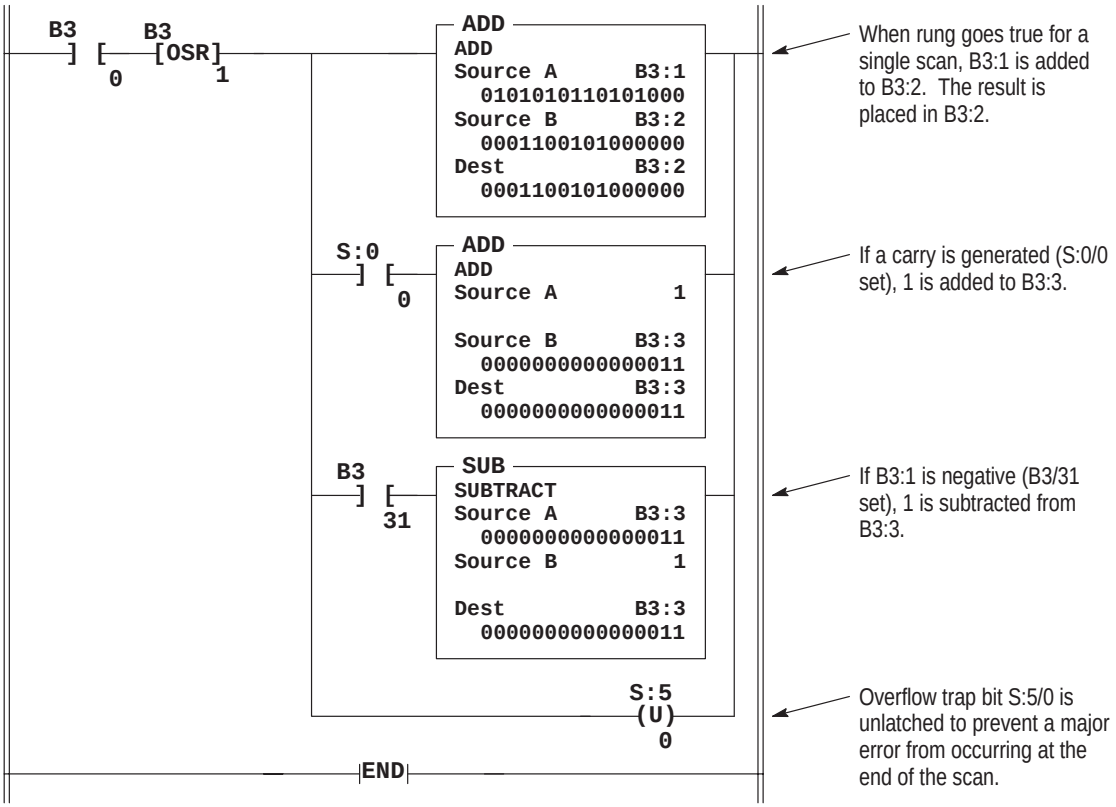
The following example shows how a 16-bit signed integer is added to a 32-bit signed integer. Remember that S:2/14 must be set for 32-bit addition.

Note that the value of the most significant 16 bits (B3:3) of the 32-bit number is increased by 1 if the carry bit S:0/0 is set and it is decreased by 1 if the number being added (B3:1) is negative.

To avoid a major error from occurring at the end of the scan, you must unlatch overflow trap bit S:5/0 as shown.

Add 16–bit value B3:1 to 32–bit value B3:2				
Add Operation		Binary	Hex	Decimal ①
Addend	B3:3 B3:2	0000 0000 0000 0011 0001 1001 0100 0000	0003 1940	203,072
Addend	B3:1	0101 0101 1010 1000	55A8	21,928
Sum	B3:3 B3:2	0000 0000 0000 0011 0110 1110 1110 1000	0003 6EE8	225,000

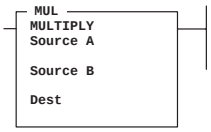
① The programming device displays 16-bit decimal values only. The decimal value of a 32-bit integer is derived from the displayed binary or hex value. For example, 0003 1940 Hex is 16^4x3 + 16^3x1 + 16^2x9 + 16^1x4 + 16^0x0 = 203,072.



Application Note: You can use the rung above with a DDV instruction and a counter to find the average value of B3:1.

Multiply (MUL)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

Use the MUL instruction to multiply one value (source A) by another (source B) and place the result in the destination.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if overflow is detected at destination; otherwise resets. On overflow, the minor error flag is also set. The value –32,768 or 32,767 is placed in the destination. Exception: If you are using an SLC 5/02 or higher processor or a MicroLogix 1000 controller and have S:2/14 (math overflow selection bit) set, then the unsigned, truncated overflow remains in the destination. For floating point destinations, the overflow result remains in the destination.
Zero (Z)	sets if result is zero; otherwise resets.
Sign (S)	sets if result is negative; otherwise resets.

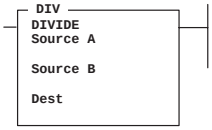
Changes to the Math Register, S:13 and S:14

Integer – Contains the 32-bit signed integer result of the multiply operation. This result is valid at overflow.

Floating Point – The math register does not change.

Divide (DIV)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

Use the DIV instruction to divide one value (source A) by another (source B). The rounded quotient is then placed in the destination. If the remainder is 0.5 or greater, round up occurs in the destination. The unrounded quotient is stored in the most significant word of the math register. The remainder is placed in the least significant word of the math register.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if division by zero or overflow is detected; otherwise resets. On overflow, the minor error flag is also set. The value 32,767 is placed in the destination. Exception: If you are using an SLC1 5/02 or higher processor or a MicroLogix 1000 controller and have S:2/14 (math overflow selection bit) set, then the unsigned, truncated overflow remains in the destination. For floating point destinations, the overflow result remains in the destination.
Zero (Z)	sets if result is zero; otherwise resets; undefined if overflow is set.
Sign (S)	sets if result is negative; otherwise resets; undefined if overflow is set.

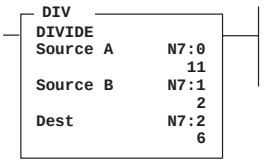
Changes to the Math Register, S:13 and S:14

Integer – The unrounded quotient is placed in the most significant word, the remainder is placed in the least significant word.

Floating Point – The math register does not change.

Example

The remainder of 11/2 is 0.5, so the quotient is rounded up to 6 and is stored in the destination. The unrounded quotient, which is 5, is stored in S:14 and the remainder, which is 1, is stored at S:13.

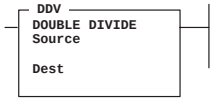


where: N7:0 = 11
N7:1 = 2
N7:2 = 6

result: S:14 = 5
S:13 = 1

Double Divide (DDV)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

The 32-bit content of the math register is divided by the 16-bit source value and the rounded quotient is placed in the destination. If the remainder is 0.5 or greater, the destination is rounded up.

This instruction typically follows a MUL instruction that creates a 32-bit result.

Updates to Arithmetic Status Bits

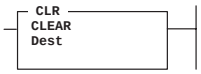
With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if division by zero or if result is greater than 32,767 or less than –32,768; otherwise resets. On overflow, the minor error flag is also set. The value 32,767 is placed in the destination.
Zero (Z)	sets if result is zero; otherwise resets.
Sign (S)	sets if result is negative; otherwise resets; undefined if overflow is set.

Changes to the Math Register, S:13 and S:14

Initially contains the dividend of the DDV operation. Upon instruction execution, the unrounded quotient is placed in the most significant word of the math register. The remainder is placed in the least significant word of the math register.

Clear (CLR)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

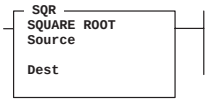
Use the CLR instruction to set the destination value of a word to zero.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	always resets.
Zero (Z)	always sets.
Sign (S)	always resets.

Square Root (SQR)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓



Output Instruction

When this instruction is evaluated as true, the square root of the absolute value of the source is calculated and the rounded result is placed in the destination.

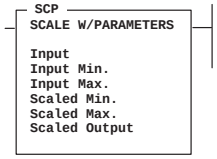
The instruction calculates the square root of a negative number without overflow or faults. In applications where the source value may be negative, use a comparison instruction to evaluate the source value to determine if the destination may be invalid.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	is reserved (integer). For floating point, it is always cleared.
Overflow (V)	always resets.
Zero (Z)	sets when destination value is zero.
Sign (S)	always resets.

Scale with Parameters (SCP)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

Use the SCP instruction to produce a scaled output value that has a linear relationship between the input and scaled values. This instruction supports integer and floating point values. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Use the following formula to convert analog input data to engineering units:

$$y = mx + b$$

Where:

y = scaled output
m = slope (scaled max. – scaled min.) / (input max. – input min.)
x = input value
b = offset (y intercept) = scaled min – (input min. × slope)

Note *The Input Minimum, Input Maximum, Scaled Minimum, and Scaled Maximum are used to determine the slope and offset values. The input value can go outside of the specified input limits and no ordering is required. For example, the scaled output value is not necessarily clamped between the scaled minimum and scaled maximum values.*

Entering Parameters

Enter the following parameters when programming this instruction:

- **Input value** can be a word address or an address of floating point data elements.
- **Input Minimum** and **Input Maximum** values determine the range of data that appears in the Input Value parameter. The value can be a word address, an integer constant, floating point data element, or a floating point constant.
- **Scaled Minimum** and **Scaled Maximum** values determine the range of data that appears in the Scaled Output parameter. The value can be a word address, an integer constant, floating point data element, or a floating point constant.
- **Scaled Output** value can be a word address or an address of floating point data elements.

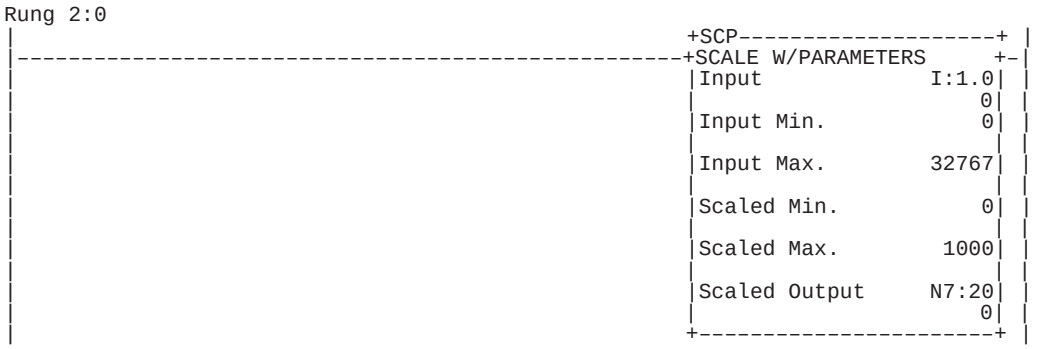
Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if overflow generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets when destination value is zero; otherwise resets.
Sign (S)	sets if the destination value is negative; otherwise resets.

Application Examples

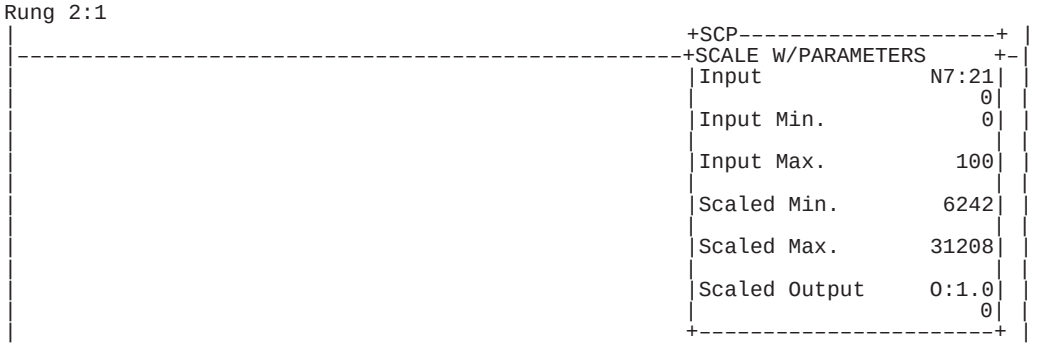
Example 1

In the first example, an analog I/O combination module (1746–NIO4I) is in slot 1 of the chassis. A pressure transducer is connected to input 0 and we want to read the value in engineering units. The pressure transducer measures pressures from 0–1000 psi and provides a 0–10V signal to the analog module. For a 0–10V signal, the analog module provides a range between 0–32,767. The following program rung places a number between 0–1000 into N7:20 based on the input signal coming from the pressure transducer into the analog module.



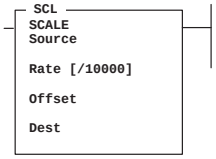
Example 2

In the second example, an analog I/O combination module (1746-NIO4I) is in slot 1 of the chassis. We want to control the proportional valve connected to output 0. The valve takes a 4–20mA signal to control how far it opens (0–100%). (Assume that additional logic is present in the program that calculates how far to open the valve in percent and places a number between 0–100 into N7:21.) The analog module provides a 4–20mA output signal for a number between 6242–31,208. The following program rung directs an analog output to provide a 4–20 mA signal to the proportional valve (N7:21), based on a number between 0–100.



Scale Data (SCL)

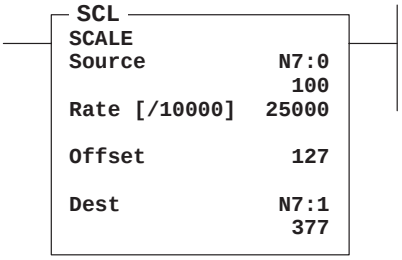
ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓



Output Instruction

When this instruction is true, the value at the source address is multiplied by the rate value. The rounded result is added to the offset value and placed in the destination.

Example



The source 100 is multiplied by 25000 and divided by 10000 and added to 127. The result 377 is placed in the destination.

Note *Anytime an underflow or overflow occurs in the destination file, minor error bit S:5/0 must be reset by the program. This must occur before the end of the current scan to prevent major error code 0020 from being declared. This instruction can overflow before the offset is added.*

Note that the term *rate* is sometimes referred to as *slope*. The rate function is limited to the range –3.2768 to 3.2767. For example, –32768/10000 to +32767/10000.

Entering Parameters

The value for the following parameters is between –32,768 to 32,767.

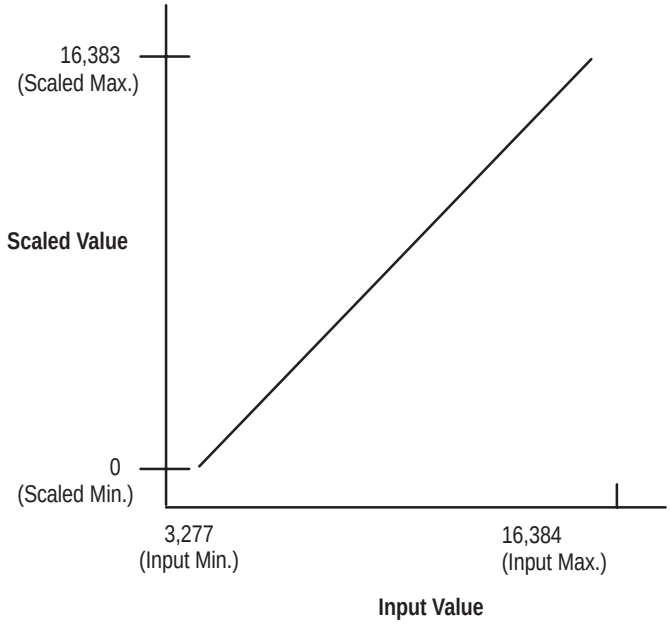
- **Source** can be either a constant or a word address.
- **Rate** (or slope) is the positive or negative value you enter divided by 10,000. It can be either a constant or a word address.
- **Offset** can be either a constant or a word address.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	is reserved.
Overflow (V)	sets if an overflow is detected; otherwise resets. On overflow, minor error bit S:5/0 is also set and the value –32,768 or 32,767 is placed in the destination. The presence of an overflow is checked before and after the offset value is applied. ①
Zero (Z)	sets when destination value is zero.
Sign (S)	sets if the destination value is negative; otherwise resets.

① If the result of the Source times the Rate, divided by 10000, is greater than 32767, the SCL instruction overflows, causing error 0020 (Minor Error Bit), and places 32767 in the Destination. This occurs regardless of the current offset.

Application Example 1 – Converting 4mA–20mA Analog Input Signal to PID Process Variable



Calculating the Linear Relationship

Use the following equations to express the linear relationship between the input value and the resulting scaled value:

Scaled value = (input value x rate) + offset

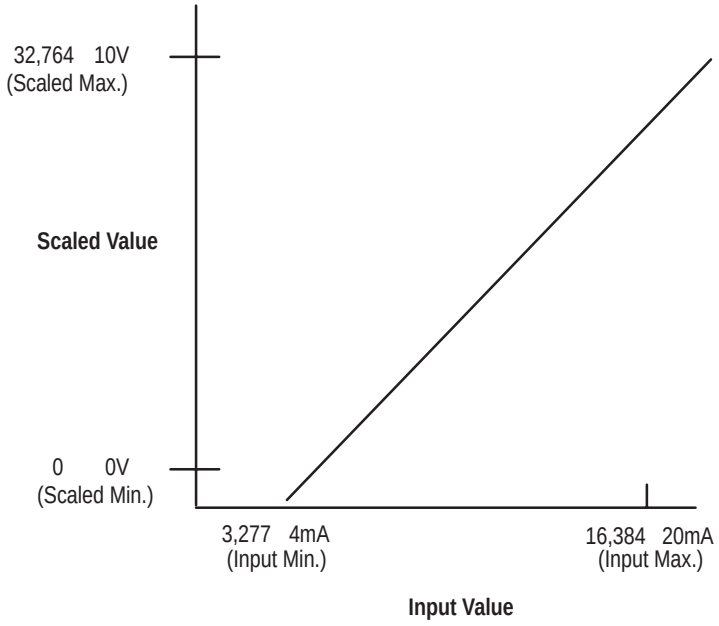
Rate = (scaled max. – scaled min.) / (input max. – input min.)

(16,383 – 0) / (16,384 – 3277) = 1.249 (or 12,490/10000)

Offset = scaled min. – (input min. x rate)

0 – (3277 × 1.249) = –4093

Application Example 2 – Scaling an Analog Input to Control an Analog Output



Calculating the Linear Relationship

Use the following equations to calculate the scaled units:

Scaled value = (input value x rate) + offset

Rate = (scaled max. – scaled min.) / (input max. – input min.)

(32,764 – 0) / (16,384 – 3277) = 2.4997 (or 24,997/10000)

Offset = scaled min. – (input min. x rate)

$0 - (3277 \times 2.4997) = -8192$

The above offset and rate values are correct for the SCL instruction. However, if the input exceeds 13,107, the instruction overflows. For example:

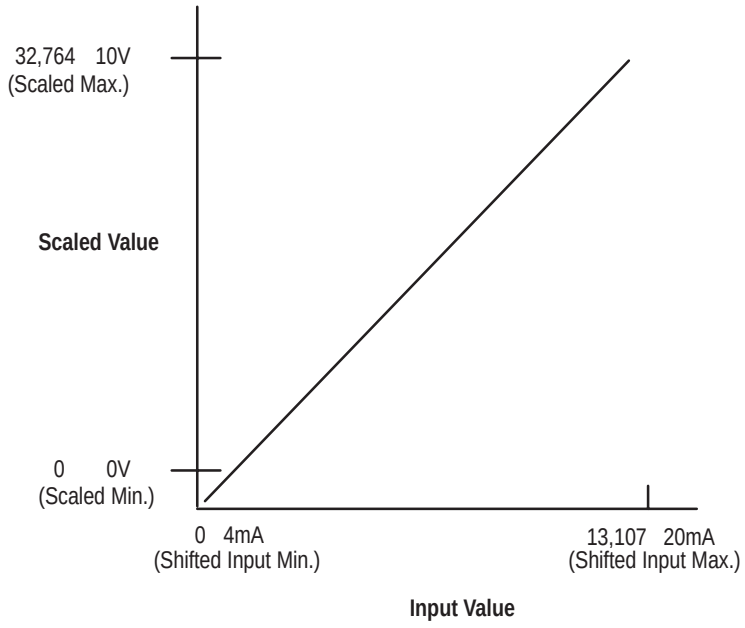
17mA = $13,926 \times 2.4997$ = 34,810 (actual overflow)

34,810 – 8192 = 26,618

Notice that an overflow occurred even though the final value was correct. This happens because the overflow condition occurred during the rate calculation.

To avoid an overflow, we recommend shifting the linear relationship along the input value axis and reduce the values.

The following graph shows the shifted linear relationship. The input minimum value of 3,277 is subtracted from the input maximum value of 16,384, resulting in the value of 13,107.



Calculating the Shifted Linear Relationship

Use the following equations to calculate the scaled units:

Scaled value = (input value x rate) + offset

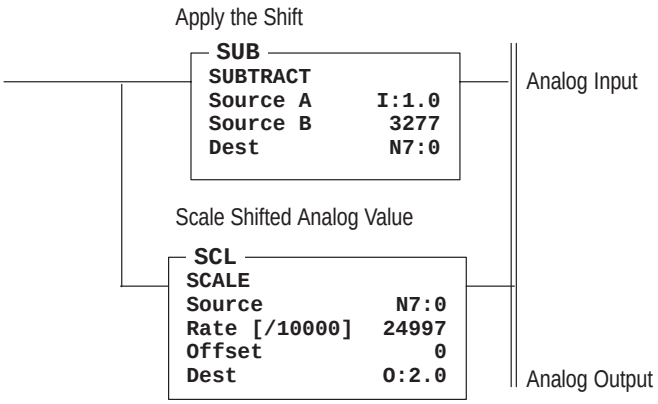
Rate = (scaled max. – scaled min.) / (input max. – input min.)

$(32{,}764 - 0) / (13{,}107 - 0) = 2.4997$ (or 24,997/10000)

Offset = scaled min. – (input min. x rate)

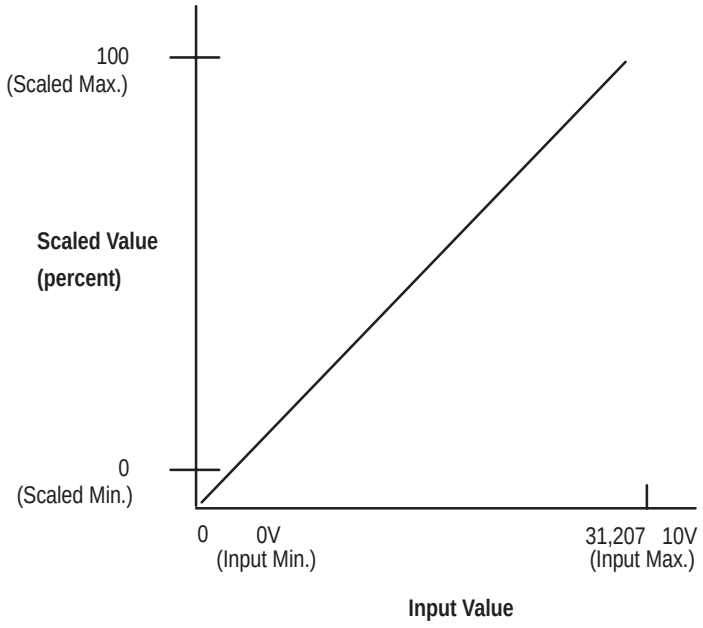
$0 - (0 \times 2.4997) = 0$

In this example, the SCL instruction is entered in the ladder logic program as follows:



Application Example 3 – Convert Voltage Input to Percent (MicroLogix)

The following example takes a 0V to 10V analog input from a MicroLogix 1000 analog controller and scales the raw input data to a value between 0 and 100%. The input value range is 0V to 10V which corresponds to 0 to 31,207 counts. The scaled value range is 0 to 100%.



Calculating the Linear Relationship

Use the following equations to calculate the scaled units:

Scaled value = (input value x rate) + offset

Rate = (scaled max. – scaled min.) / (input max. – input min.)

= (100 – 0) / (31,207 – 0)

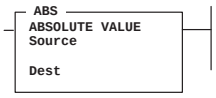
= .00320 (or 320/10000)

Offset = scaled min. – (input min. x rate)

= 0 – (0 × .00320) = 0

Absolute (ABS)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

Use the ABS instruction to calculate the absolute value of the Source and place the result in the Destination. This instruction supports integer and floating point values. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Entering Parameters

Enter the following parameters when programming this instruction:

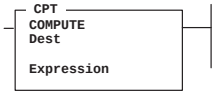
- **Source** can be a word address, an integer constant, floating point data element, or a floating point constant.
- **Destination** can only be a word address or a floating point data element.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	always resets with a floating point value; sets if the input is –32,768 (integer value).
Zero (Z)	sets when destination value is zero; otherwise resets.
Sign (S)	always resets.

Compute (CPT)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

The CPT instruction performs copy, arithmetic, logical, and conversion operations. You define the operation in the Expression and the result is written in the Destination. The CPT uses functions to operate on one or more values in the Expression to perform operations such as:

- converting from one number format to another
- manipulating numbers
- performing trigonometric functions

Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Instructions that can be used in the Expression include:
+, –, *, | (DIV), SQR, – (NEG), NOT, XOR, OR, AND, TOD, FRD, LN, TAN, ABS, DEG, RAD, SIN, COS, ATN, ASN, ACS, LOG, and ** (XPY).

Note The execution time of a CPT instruction is longer than a single arithmetic operation and uses more instruction words.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Destination** can be a word address or the address of a floating-point data element.
- **Expression** is zero or more lines, with up to 28 characters per line, up to 255 characters.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	sets based on the result of the last instruction in the Expression.
Overflow (V)	sets any time an overflow occurs during the evaluation of the Expression.
Zero (Z)	sets based on the result of the last instruction in the Expression.
Sign (S)	sets based on the result of the last instruction in the Expression.

The above bits are cleared at the start of the CPT instruction. See S:34/2 for special handling of the math status bits when using floating point.

Application Example

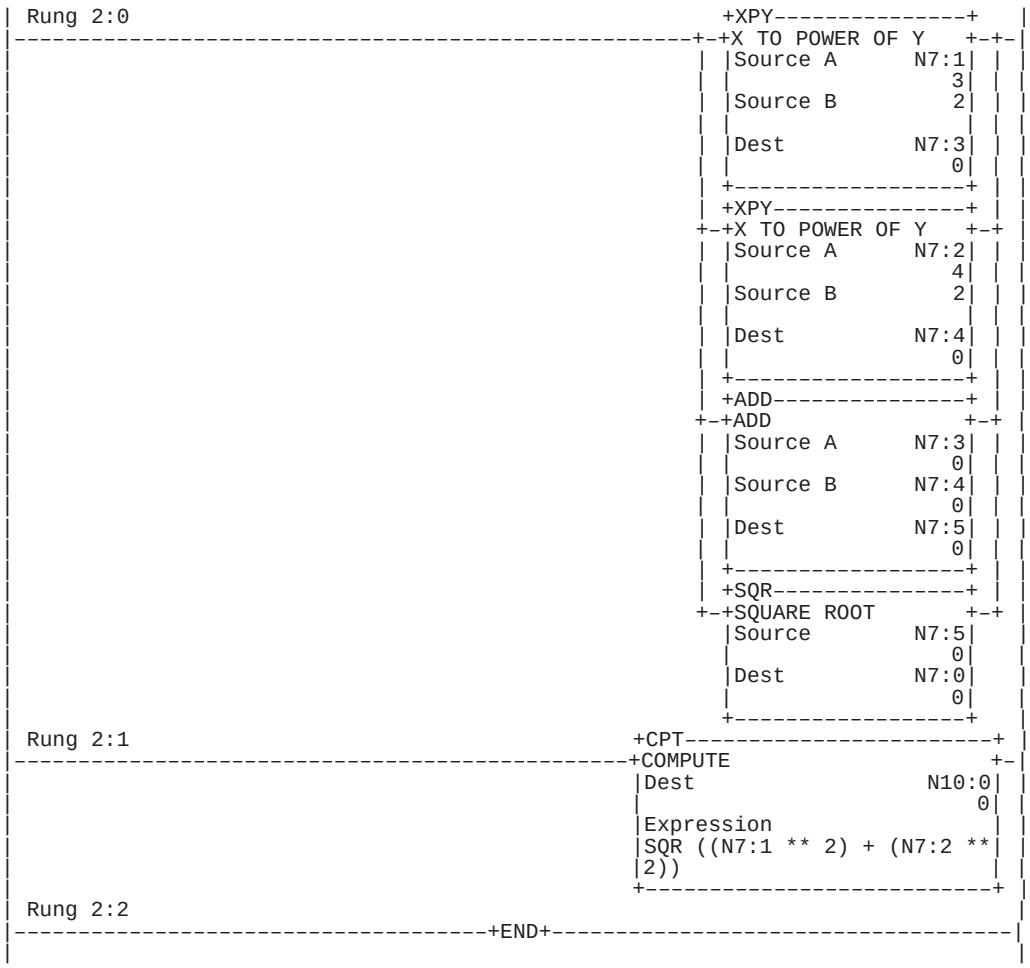
This application example uses Pythagorean's theorem to find the length of the long leg of a triangle, knowing the two other leg lengths. Use the following equation:

$$c^2 = a^2 + b^2$$

$$\text{where } c = \sqrt{(a^2 + b^2)}$$

$$N10:0 = \sqrt{(N7:1)^2 + (N7:2)^2}$$

Rung 2:0 uses standard math instructions to implement Pythagorean's theorem. Rung 2:1 uses the CPT instruction to obtain the same calculation.



Swap (SWP)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

Output Instruction

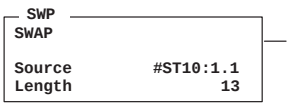
Use this instruction to swap the low and high bytes of a specified number of words in a bit, integer, ASCII, or string file. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** can only be an indexed word address.
- **Length** refers to the number of words to be swapped, regardless of the file type. The address is limited to integer constants. For bit, integer, and ASCII file types, the length range is 1 to 128. For the string file type, the length range is 1 to 41. Note that this instruction is restricted to a single string element and cannot cross a string element boundary.

The following example shows how the SWP instruction works.

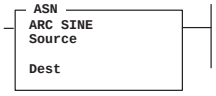


Before:
ST10:1 = abcdefghijklmnopqrstuvwxyz

After
ST10:1 = badcfehgjilknmporqtsvuxwzy

Arc Sine (ASN)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

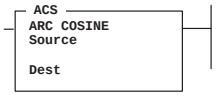
Use the ASN instruction to take the arc sine of a number (source in radians) and store the result (in radians) in the destination. The source must be greater than or equal to –1 and less than or equal to 1. The resulting value in the destination is always greater than or equal to –Pi/2 and less than or equal to Pi/2, where Pi = 3.141592. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

Arc Cosine (ACS)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

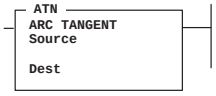
Use the ACS instruction to take the arc cosine of a number (source in radians) and store the result (in radians) in the destination. The source must be greater than or equal to –1 and less than or equal to 1. The resulting value in the destination is always greater than or equal to 0 and less than or equal to Pi, where Pi = 3.141592. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	always resets.

Arc Tangent (ATN)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

Use the ATN instruction to take the arc tangent of a number (source) and store the result (in radians) in the destination. The resulting value in the destination is always greater than or equal to –Pi/2 and less than or equal to Pi/2, where Pi = 3.141592. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

Cosine (COS)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

Output Instruction

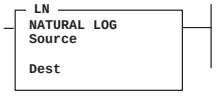
Use the COS instruction to take the cosine of a number (source in radians) and store the result in the destination. The source must be greater than or equal to –205887.4 and less than or equal to 205887.4. The greatest accuracy is achieved when the source is greater than –2 Pi and less than 2 Pi, where Pi = 3.141592. The resulting value in the destination is always greater than or equal to –1 and less than or equal to 1. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

Natural Log (LN)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

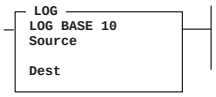
Use the LN instruction to take the natural log of the value in the source and store the result in the destination. The source must be greater than zero. The resulting value in the destination is always greater than or equal to –87.33654 and less than or equal to 88.72284. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

Log to the Base 10 (LOG)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

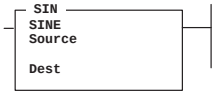
Use the LOG instruction to take the log base 10 of the value in the source and store the result in the destination. The source must be greater than zero. The resulting value in the destination is always greater than or equal to –37.92978 and less than or equal to 38.53184. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

Sine (SIN)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

Use the SIN instruction to take the sine of a number (source in radians) and store the result in the destination. The source must be greater than or equal to –205887.4 and less than or equal to 205887.4. The greatest accuracy is achieved when the source is greater than –2 Pi and less than 2 Pi, where Pi = 3.141592. The resulting value in the destination is always greater than or equal to –1 and less than or equal to 1. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

Tangent (TAN)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

Output Instruction

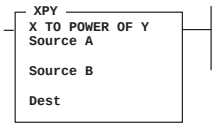
Use the TAN instruction to take the tangent of a number (source in radians) and store the result in the destination. The value in the source must be greater than or equal to –102943.7 and less than or equal to 102943.7. The greatest accuracy is achieved when the source is greater than –2 Pi and less than 2 Pi, where Pi = 3.141592. The resulting value in the destination is either a real number or infinity. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

X to the Power of Y (XPY)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

Use the XPY instruction to raise a value (source A) to a power (source B) and store the result in the destination. If the value in source A is negative, the exponent (source B) should be a whole number. If it is not a whole number, the overflow bit is set and the absolute value of the base is used in the calculation. Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

The XPY instruction uses the following algorithm:

$$XPY = 2 ** (Y * \log_2 (X))$$

If any of the intermediate operations in this algorithm produce an overflow, the Arithmetic Overflow Status bit (S:0/1) is set.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if an overflow is generated or an unsupported input is detected; otherwise resets.
Zero (Z)	sets if the result is zero; otherwise resets.
Sign (S)	sets if the result is negative; otherwise resets.

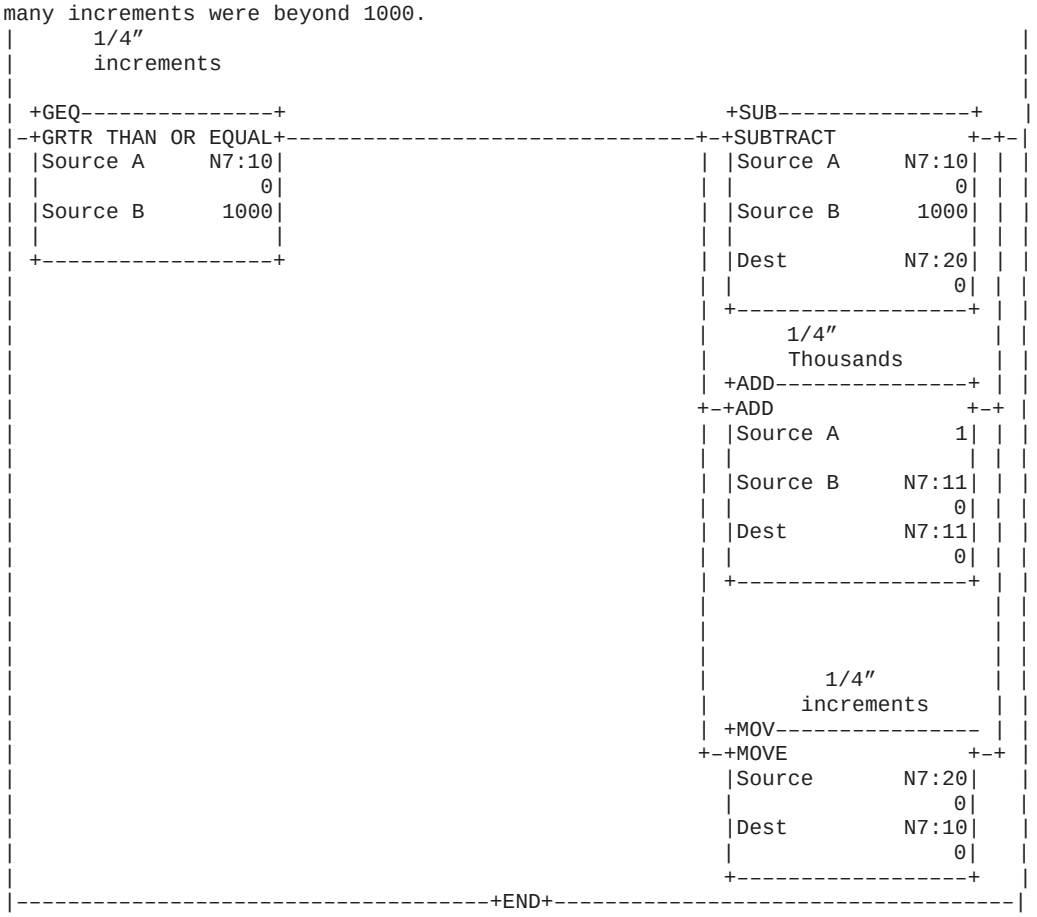
Math Instructions in the Paper Drilling Machine Application Example

This section provides ladder rungs to demonstrate the use of math instructions. The rungs are part of the paper drilling machine application example described in appendix H. You will be adding to the subroutine in file 7 that was started in chapter 1.

Adding File 7

```
Rung 7:1
This rung resets the number of 1/4″ increments and the 1/4″ thousands when the
″drill change reset″ keyswitch is energized. This should occur following each
drill bit change.
| drill                                                    1/4″             |
| change                                                   Thousands        |
| reset                                                                     |
| keyswitch                                                                 |
|   I:1.0                                            +CLR---------------+   |
|----] [---------------------------------------------+-+CLEAR             +-+-|
|       8                                           | |Dest         N7:11| | |
|                                                   | |                 0| | |
|                                                   | +------------------+ | |
|                                                   |      1/4″            | |
|                                                   |      increments      | |
|                                                   |                      | |
|                                                   | +CLR---------------+ | |
|                                                   +-+CLEAR             +-+ |
|                                                     |Dest         N7:10|   |
|                                                     |                 0|   |
|                                                     +------------------+   |
Rung 7:6
Keep a running total of how many inches of paper have been drilled with the
current drill bit.  Every time a hole is drilled, add the thickness (in 1/4″s)
to the running total (kept in 1/4″s).  The OSR is necessary because the ADD
executes every time the rung is true, and the drill body would actuate the
DRILL DEPTH limit switch for more than 1 program scan.  Integer N7:12 is the
integer–converted value of the BCD thumbwheel on inputs I:3/11 – I:3/14.
| Drill     |Tool Wear                                             1/4″          |
| Depth LS  |  OSR 1                                               increments    |
|                                                                             |
|   I:1.0        B3:1                                  +ADD---------------+ |
|----] [-------[OSR]-----------------------------------+ADD               +-|
|       4          8                                   |Source A     N7:12| |
|                                                      |                 1| |
|                                                      |Source B     N7:10| |
|                                                      |                 0| |
|                                                      |Dest         N7:10| |
|                                                      |                 0| |
|                                                      +------------------+ |
```

Rung 7:7
When the number of 1/4″ increments surpasses 1000, find out now many increments we are past 1000 and store in N7:20, add 1 to the total of ’1000 1/4″’ increments, and re–initialize the 1/4″ increments accumulator to how many increments were beyond 1000.



4 Data Handling Instructions

This chapter contains general information about the data handling instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the data handling instructions in use.

Data Handling Instructions

Instruction		Purpose	Page
Mnemonic	Name		
TOD	Convert to BCD	Converts the integer source value to BCD format and stores it in the destination.	**4–3**
FRD	Convert from BCD	Converts the BCD source value to an integer and stores it in the destination.	**4–6**
DEG	Convert from Radians to Degrees	Converts radians (source) to degrees and stores the result in the destination.	**4–10**
RAD	Convert from Degrees to Radians	Converts degrees (source) to radians and stores the result in the destination.	**4–11**
DCD	Decode 4 to 1 of 16	Decodes a 4-bit value (0 to 15), turning on the corresponding bit in the 16-bit destination.	**4–12**
ENC	Encode 1 of 16 to 4	Encodes a 16-bit source to a 4-bit value. Searches the source from the lowest to the highest bit, and looks for the first set bit. The corresponding bit position is written to the destination as an integer.	**4–13**
COP and FLL	Copy File and Fill File	The COP instruction copies data from the source file to the destination file The FLL instruction loads a source value into each position in the destination file.	**4–14**

continued on next page

Instruction		Purpose	Page
Mnemonic	**Name**		
MOV	Move	Moves the source value to the destination.	**4–19**
MVM	Masked Move	Moves data from a source location to a selected portion of the destination.	**4–20**
AND	And	Performs a bitwise AND operation.	**4–22**
OR	Or	Performs a bitwise inclusive OR operation.	**4–23**
XOR	Exclusive Or	Performs a bitwise exclusive OR operation.	**4–24**
NOT	Not	Performs a NOT operation.	**4–25**
NEG	Negate	Changes the sign of the source and stores it in the destination.	**4–26**
FFL and FFU	FIFO Load and FIFO Unload	The FFL instruction loads a word into a FIFO stack on successive false-to-true transitions. The FFU unloads a word from the stack on successive false-to-true transitions. The first word loaded is the first to be unloaded.	**4–29**
LFL and LFU	LIFO Load and LIFO Unload	The LFL instruction loads a word into a LIFO stack on successive false-to-true transitions. The LFU unloads a word from the stack on successive false-to-true transitions. The last word loaded is the first to be unloaded.	**4–30**

About the Data Handling Instructions

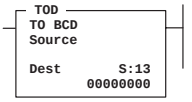
Use these instructions to convert information, manipulate data in the controller, and perform logic operations.

In this chapter you will find a general overview preceding groups of instructions. Before you learn about the instructions in each of these groups, we suggest that you read the overview. This chapter contains the following overviews:

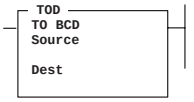
- Move and Logical Instructions Overview
- FIFO and LIFO Instructions Overview

Convert to BCD (TOD)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction
Fixed and SLC 5/01
Processors



Output Instruction
SLC 5/02 and higher processors
and MicroLogix 1000 controllers

Use this instruction to convert 16-bit integers into BCD values.

With Fixed and SLC 5/01 processors, the destination can only be the math register. With SLC 5/02 and higher processors and MicroLogix 1000 controllers, the destination parameter can be a word address in any data file, or it can be the math register, S:13 and S:14.

If the integer value you enter is negative, the absolute value of the number is used for conversion.

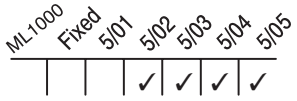
Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if the BCD result is larger than 9999. Overflow results in a minor error.
Zero (Z)	sets if destination value is zero.
Sign (S)	sets if the source word is negative; otherwise resets.

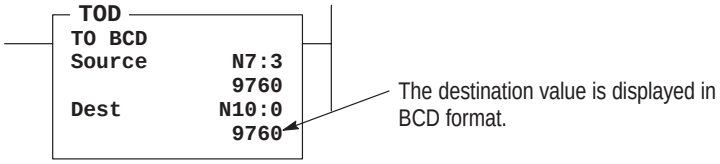
Changes to the Math Register, S:13 and S:14

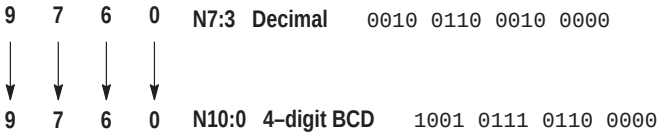
Contains the 5–digit BCD result of the conversion. This result is valid at overflow.

Example 1

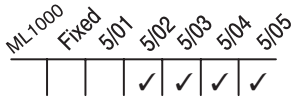


The integer value 9760 stored at N7:3 is converted to BCD and the BCD equivalent is stored in N10:0. The maximum BCD value possible is 9999.



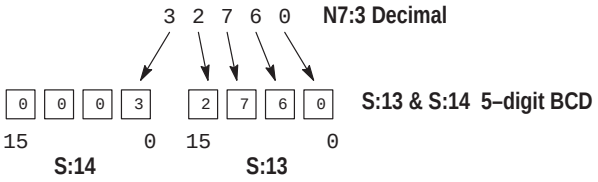


Example 2

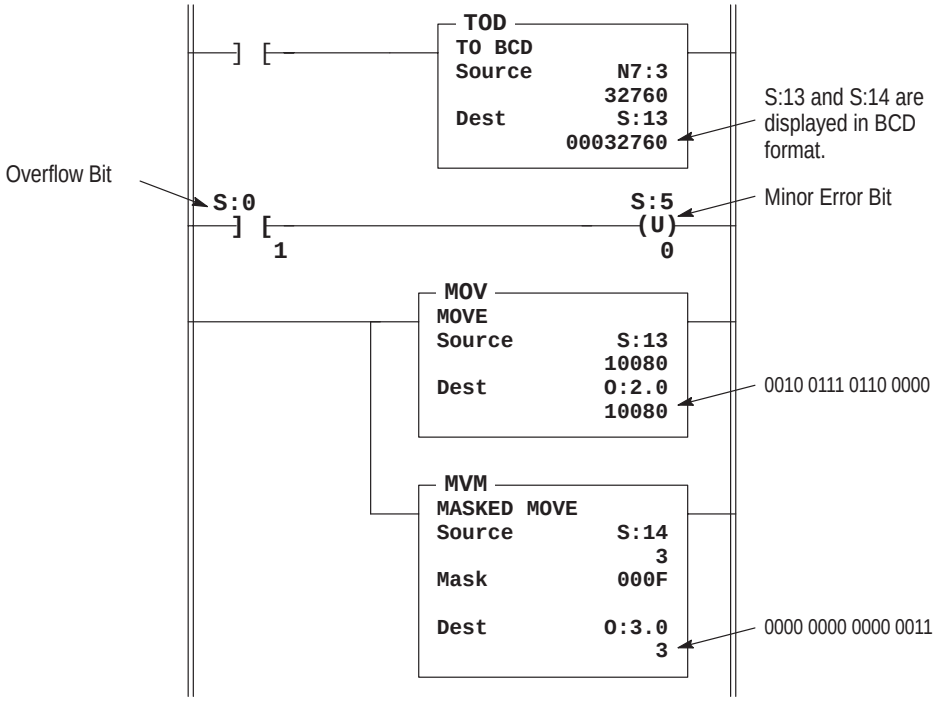


The integer value 32760 stored at N7:3 is converted to BCD. The 5-digit BCD value is stored in the math register. The lower 4 digits of the BCD value is moved to output word O:2 and the remaining digit is moved through a mask to output word O:3.

When using the math register as the destination parameter in the TOD instruction, the maximum BCD value possible is 32767. However, for BCD values above 9999, the overflow bit is set, resulting in minor error bit S:5/0 also being set. Your ladder program can unlatch S:5/0 before the end of the scan to avoid major error 0020, as done in this example.

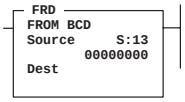


This example will output the absolute value (0–32767) contained in N7:3 as 5 BCD digits in output slots 2 and 3.



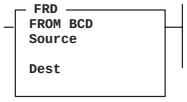
Convert from BCD (FRD)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction
Fixed and SLC 5/01
Processors

Use this instruction to convert BCD values to integer values. With Fixed and SLC 5/01 processors, the source can only be the math register. With SLC 5/02 and higher processors and MicroLogix 1000 controllers, the source parameter can be a word address in any data file, or it can be the math register, S:13.



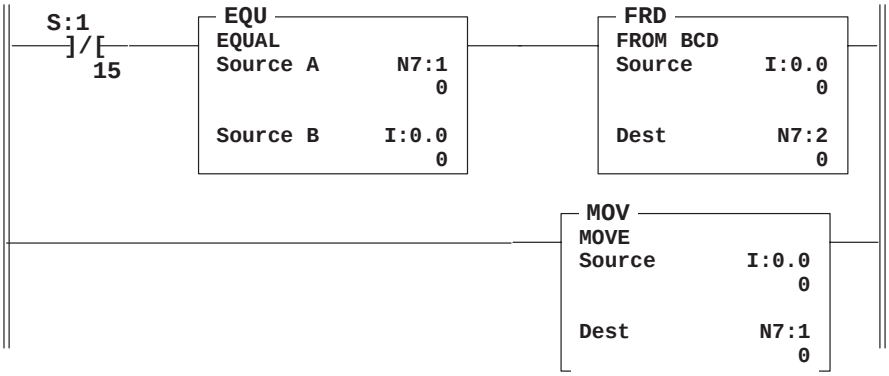
Output Instruction
SLC 5/02 and higher Processors
and MicroLogix 1000 controllers

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if non–BCD value is contained at the source or the value to be converted is greater than 32,767; otherwise reset. Overflow results in a minor error.
Zero (Z)	sets if destination value is zero.
Sign (S)	always resets.

Note

We recommend that you always provide ladder logic filtering of all BCD input devices prior to performing the FRD instruction. The slightest difference in point-to-point input filter delay can cause the FRD instruction to overflow due to the conversion of a non–BCD digit.



In the above example, the two rungs cause the processor to verify that the value at I:0.0 remains the same for two consecutive scans before it executes the FRD. This prevents the FRD from converting a non-BCD value during an input value change.

Note *To convert numbers larger than 9999 BCD, the source must be the Math Register (S:13). You must reset the Minor Error bit (S:5.0) to prevent an error.*

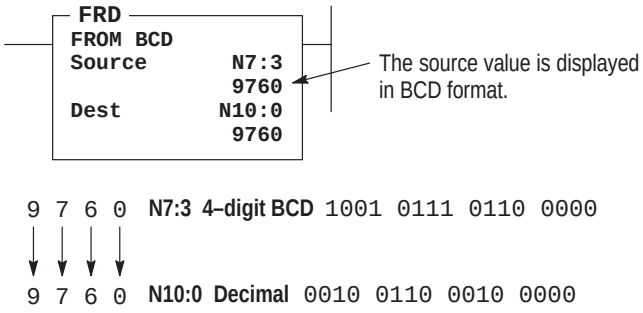
Changes to the Math Register, S:13 and S:14

Used as the source for converting the entire number range of a register.

Example 1

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓

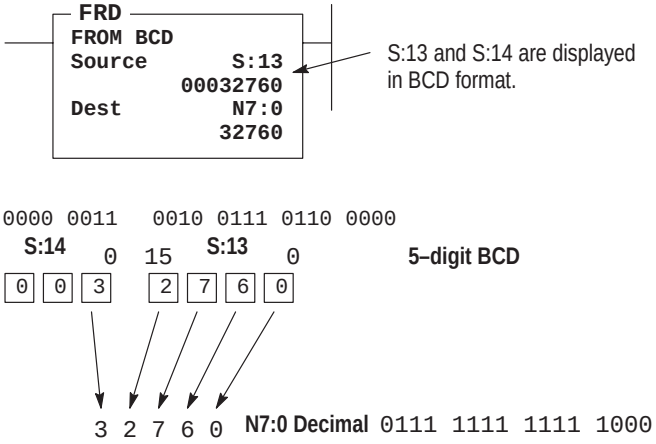
The BCD value 9760 at source N7:3 is converted and stored in N10:0. The maximum source value is 9999, BCD.



Example 2

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓

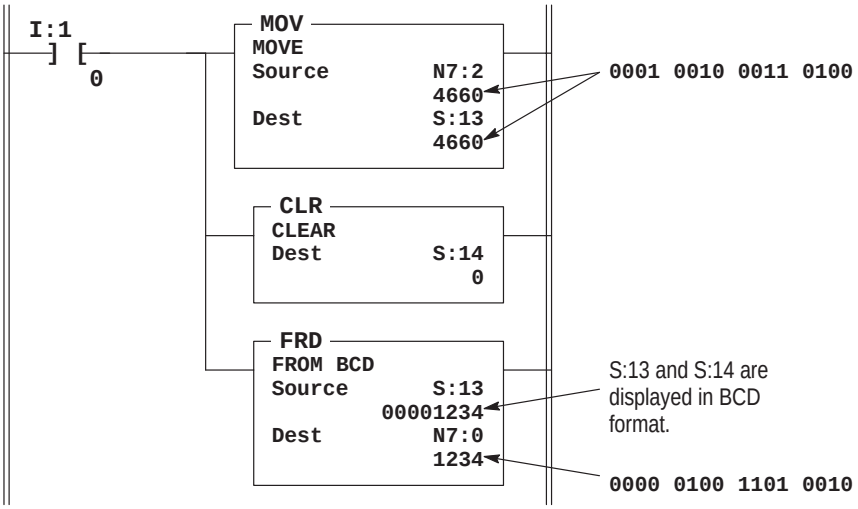
The BCD value 32760 in the math register is converted and stored in N7:0. The maximum source value is 32767, BCD.



You should convert BCD values to integer before you manipulate them in your ladder program. If you do not convert the values, the processor manipulates them as integers and their value is lost.

Note *If the math register (S:13 and S:14) is used as the source for the FRD instruction and the BCD value does not exceed 4 digits, be sure to clear word S:14 before executing the FRD instruction. If S:14 is not cleared and a value is contained in this word from another math instruction located elsewhere in the program, an incorrect decimal value will be placed in the destination word.*

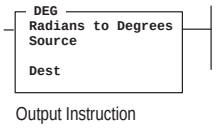
Clearing S:14 before executing the FRD instruction is shown below:



When the input condition is set (1), a BCD value (transferred from a 4-digit thumbwheel switch for example) is moved from word N7:2 into the math register. Status word S:14 is then cleared to make certain that unwanted data is not present when the FRD instruction is executed.

Radian to Degrees (DEG)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓



Output Instruction

Use this instruction to convert radians (source) to degrees and store the result in the destination. The following formula applies:

Source × 180/Π
where Π = 3.141592

Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Entering Parameters

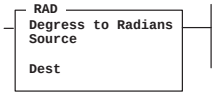
- **Source** is the integer and/or floating point values.
- **Destination** is the address of the word where the data is to be stored.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if overflow generated or an unsupported input is detected; otherwise resets
Zero (Z)	sets if the result is zero; otherwise resets
Sign (S)	sets if the result is negative; otherwise resets

Degrees to Radians (RAD)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

Use this instruction to convert degrees (source) to radians and store the result in the destination. The following formula applies:

Source × Π/180
where Π = 3.141592

Use this instruction with SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors processors.

Entering Parameters

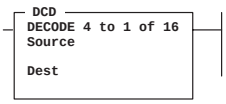
- **Source** is the integer and/or floating point values.
- **Destination** is the address of the word where the data is to be stored.

Updates to Arithmetic Status Bits

With this Bit:	The Processor:
Carry (C)	always resets.
Overflow (V)	sets if overflow generated or an unsupported input is detected; otherwise resets
Zero (Z)	sets if the result is zero; otherwise resets
Sign (S)	sets if the result is negative; otherwise resets

Decode 4 to 1 of 16 (DCD)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

When executed, this instruction sets one bit of the destination word. The particular bit that is turned on depends on the value of the first four bits of the source word. See the table below.

Use this instruction to multiplex data in applications such as rotary switches, keypads, and bank switching.

	Source				Destination															
Bit	15–04	03	02	01	00	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	x	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	x	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	x	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	x	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	x	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	x	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	x	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	x	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	x	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	x	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	x	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Entering Parameters

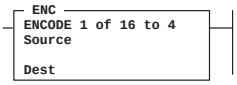
- **Source** is the address that contains the bit decode information. Only the first four bits (0–3) are used by the DCD instruction. The remaining bits may be used for other application specific needs. Change the value of the first four bits of this word to select one bit of the destination word.
- **Destination** is the address of the word where the data is to be stored.

Updates to Arithmetic Status Bits

Unaffected.

Encode 1 of 16 to 4 (ENC)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓						



Output Instruction

When the rung is true, this output instruction searches the source from the lowest to the highest bit, and looks for the first set bit. The corresponding bit position is written to the destination as an integer as shown in the table below.

	Source																Destination				
Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	15–04	03	02	01	00
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	0	0	0	0
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	x	0	0	0	1
	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	x	0	0	1	0
	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	x	0	0	1	1
	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	x	0	1	0	0
	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	0	x	0	1	0	1
	x	x	x	x	x	x	x	x	x	x	1	0	0	0	0	0	x	0	1	1	0
	x	x	x	x	x	x	x	x	x	1	0	0	0	0	0	0	x	0	1	1	1
	x	x	x	x	x	x	x	x	1	0	0	0	0	0	0	0	x	1	0	0	0
	x	x	x	x	x	x	x	1	0	0	0	0	0	0	0	0	x	1	0	0	1
	x	x	x	x	x	x	1	0	0	0	0	0	0	0	0	0	x	1	0	1	0
	x	x	x	x	x	1	0	0	0	0	0	0	0	0	0	0	x	1	0	1	1
	x	x	x	x	1	0	0	0	0	0	0	0	0	0	0	0	x	1	1	0	0
	x	x	x	1	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	0	1
	x	x	1	0	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	1	0
	x	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	1	1
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0					

Entering Parameters

- **Source** is the address of the word to be encoded. Only one bit of this word should be on at any time. If more than one bit in the source is set, the destination bits are set based on the least significant bit that is set. If a source of zero is used, all of the destination bits are reset and the zero bit is set.
- **Destination** is the address that contains the bit encode information. Bits 4–15 of the destination are reset by the ENC instruction.

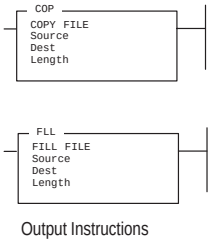
Updates to Arithmetic Status Bits

The arithmetic status bits are found in Word 0, bits 0–3 in the status file. After an instruction is executed, the arithmetic status bits in the status file are updated:

With this Bit:		The Controller:
S:0/0	Carry (C)	always resets.
S:0/1	Overflow (V)	sets if more than one bit in the source is set; otherwise reset. The math overflow bit (S:5/0) is *not* set.
S:0/2	Zero (Z)	sets if destination value is zero.
S:0/3	Sign (S)	always resets.

Copy File (COP) and Fill File (FLL) Instructions

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



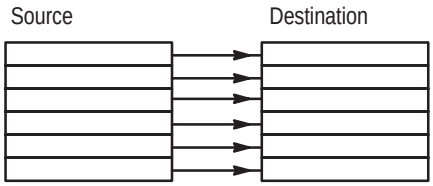
Output Instructions

The destination file type determines the number of words that an instruction transfers. For example, if the destination file type is a counter and the source file type is an integer, three integer words are transferred for each element in the counter-type file.

After a COP or FLL instruction is executed, index register S:24 is cleared to zero.

Using COP

This instruction copies blocks of data from one location into another. It uses no status bits. If you need an enable bit, program an output instruction (OTE) in parallel using an internal bit as the output address. The following figure shows how file instruction data is manipulated.



File to File

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is the address of the file you want to copy. You must use the file indicator (#) in the address. When using either an SLC 5/03 (OS301 or higher), SLC 5/04 (OS401), or SLC 5/05 processor, floating point and string values are supported.
- **Destination** is the starting address where the instruction stores the copy. You must use the file indicator (#) in the address. When using either an SLC 5/03 (OS301 or higher), SLC 5/04 (OS401), or SLC 5/05 processor, floating point and string values are supported.

- **Length** is the number of *elements* in the file you want to copy.
  - For SLC processors, if the destination file type is 3 words per element (Timer or Counter), you can specify a maximum length of 42. If the destination file type is 1 word per element, you can specify a maximum length of 128 words.
  - MicroLogix 1000 controllers, see the table below:

If the destination file type is a:	then you can specify a maximum length of:	
	Discrete Controllers	Analog Controllers
Output	1	5
Input	2	8
Status	33	—
Bit	32	—
Timer	40	—
Counter	32	—
Control	16	—
Integer	105	

Note *The maximum lengths apply when the source is of the same file type.*

All elements are copied from the source file into the destination file each time the instruction is executed. Elements are copied in ascending order.

If your destination file type is a timer, counter, or control file, be sure that the source words corresponding to the status words of your destination file contains zeros.

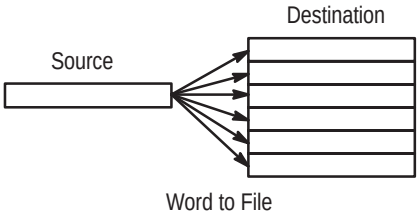
Be sure that you accurately specify the starting address and length of the data block you are copying. The instruction will not write over a file boundary (such as between files N16 and N17) at the destination. An error occurs if a write is attempted over a file boundary.

You can perform file shifts by specifying a source element address one or more elements greater than the destination element address within the same file. This shifts data to lower element addresses.

Using FLL

This instruction loads elements of a file with either a program constant or value from an element address.

The instruction fills the words of a file with a source value. It uses no status bits. If you need an enable bit, program a parallel output that uses a storage address. The following figure shows how file instruction data is manipulated.



Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is the program constant or element address. The file indicator (#) is *not* required for an element address. When using either an SLC 5/03 (OS301 or higher), SLC 5/04 (OS401), or SLC 5/05 processor, floating point and string values are supported.
- **Destination** is the destination starting address of the file you want to fill. You must use the file indicator (#) in the address. When using either an SLC 5/03 (OS301 or higher), SLC 5/04 (OS401), or SLC 5/05 processor, floating point and string values are supported.

- **Length** is the number of *elements* in the file you want filled.

  – For SLC processors, if the destination file type is 3 words per element (Timer or Counter), you can specify a maximum length of 42. If the destination file type is 1 word per element, you can specify a maximum length of 128 words.

  – For MicroLogix 1000 controllers, see the table below:

If the destination file type is a:	then you can specify a maximum length of:	
	Discrete Controllers	Analog Controllers
Output	1	5
Input	2	8
Status	33	—
Bit	32	—
Timer	40	—
Counter	32	—
Control	16	—
Integer	105	

All elements are filled from the source value (typically a constant) into the specified destination file each scan the rung is true. Elements are filled in ascending order.

The instruction will not write over a file boundary (such as between files N16 and N17) at the destination. An error is declared if a write is attempted over a file boundary.

Move and Logical Instructions Overview

The following general information applies to move and logical instructions.

Entering Parameters

- **Source** is the address of the value on which the logical or move operation is to be performed. The source can be a word address or a program constant, unless otherwise described. If the instruction has two source operands, it does not accept program constants in both operands.

  When using either an SLC 5/03 (OS301 or higher), SLC 5/04, or SLC 5/05 processor, floating point and string values are supported.

- **Destination** is the result address of a move or logical operation. It must be a word address.

Using Indexed Word Addresses

You have the option of using indexed word addresses for instruction parameters specifying word addresses. Indexed addressing is discussed in appendix C.

Updates to Arithmetic Status Bits

The arithmetic status bits are found in Word 0, bits 0–3 in the controller status file. After an instruction is executed, the arithmetic status bits in the status file are updated.

Using Indirect Word Addresses

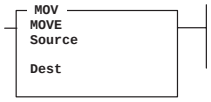
You have the option of using indirect word-level and bit-level addresses for instructions specifying word addresses when using an SLC 5/03 (OS302), SLC 5/04 (OS401), or SLC 5/05 processors processors. See appendix C for more information.

Changes to the Math Register, S:13 and S:14

Move and logical instructions do not affect the math register.

Move (MOV)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

This output instruction moves the source value to the destination location. As long as the rung remains true, the instruction moves the data each scan.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is the address or constant of the data you want to move.
- **Destination** is the address where the instruction moves the data.

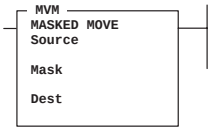
Application Note: If you wish to move one word of data without affecting the math flags, use a copy (COP) instruction with a length of 1 word instead of the MOV instruction.

Updates to Arithmetic Status Bits

With this Bit:		The Controller:
S:0/0	Carry (C)	always resets.
S:0/1	Overflow (V)	always resets.
S:0/2	Zero (Z)	sets if result is zero; otherwise resets.
S:0/3	Sign (S)	sets if result is negative (most significant bit is set); otherwise resets.

Masked Move (MVM)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

The MVM instruction is a word instruction that moves data from a source location to a destination, and allows portions of the destination data to be masked by a separate word. As long as the rung remains true, the instruction moves the data each scan.

Entering Parameters

Enter the following parameters when programming this instruction:

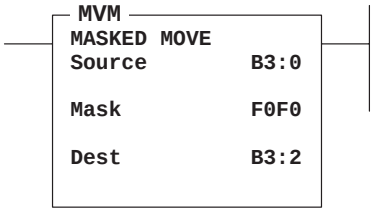
- **Source** is the address of the data you want to move.
- **Mask** is the address of the mask through which the instruction moves data; the mask can be a hexadecimal value (constant).
- **Destination** is the address where the instruction moves the data.

Updates to Arithmetic Status Bits

With this Bit:		The Controller:
S:0/0	Carry (C)	always resets.
S:0/1	Overflow (V)	always resets.
S:0/2	Zero (Z)	sets if result is zero; otherwise resets.
S:0/3	Sign (S)	sets if result is negative; otherwise resets.

Operation

When the rung containing this instruction is true, data at the source address passes through the mask to the destination address. See the figure below.



B3:2 before move

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

source B3:0

0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

Mask F0F0

1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0

B3:2 after move

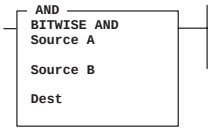
0	1	0	1	1	1	1	1	0	1	0	1	1	1	1	1

Mask data by resetting bits in the mask; pass data by setting bits in the mask to one. The bits of the mask can be fixed by a constant value, or you can vary them by assigning the mask a direct address.

Note *Bits in the destination that correspond to zeros in the mask are not altered.*

And (AND)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

This instruction performs a bit-by-bit logical AND. The operation is performed using the value at source A and the value at source B. The result is stored in the destination.

Truth Table

Dest = A AND B		
A	**B**	**Dest**
0	0	0
1	0	0
0	1	0
1	1	1

Source A and B can either be a word address or a constant; however, both sources cannot be a constant. The destination must be a word address.

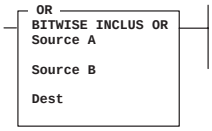
Application Note: When entering constants, you can use the ampersand (&) operator to change the radix of your entry. For example, instead of entering –1 as a constant, you could enter &B1111111111111111 or &HFFFF.

Updates to Arithmetic Status Bits

With this Bit:		The Controller:
S:0/0	Carry (C)	always resets.
S:0/1	Overflow (V)	always resets.
S:0/2	Zero (Z)	sets if result is zero; otherwise resets.
S:0/3	Sign (S)	sets if most significant bit is set; otherwise resets.

Or (OR)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

This instruction performs a bit-by-bit logical OR. The operation is performed using the value at source A and the value at source B. The result is stored in the destination.

Truth Table

Dest = A OR B		
A	**B**	**Dest**
0	0	0
1	0	1
0	1	1
1	1	1

Source A and B can either be a word address or a constant; however, both sources cannot be a constant. The destination must be a word address.

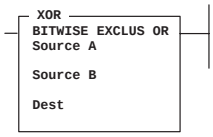
Application Note: When entering constants, you can use the ampersand (&) operator to change the radix of your entry. For example, instead of entering –1 as a constant, you could enter &B1111111111111111 or &HFFFF.

Updates to Arithmetic Status Bits

With this Bit:		The Controller:
S:0/0	Carry (C)	always resets.
S:0/1	Overflow (V)	always resets.
S:0/2	Zero (Z)	sets if result is zero; otherwise resets.
S:0/3	Sign (S)	sets if result is negative (most significant bit is set) otherwise resets.

Exclusive Or (XOR)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

This instruction performs a bit-by-bit logical XOR. The operation is performed using the value at source A and the value at source B. The result is stored in the destination.

Truth Table

Dest = A XOR B		
A	**B**	**Dest**
0	0	0
1	0	1
0	1	1
1	1	0

Source A and B can either be a word address or a constant; however, both sources cannot be a constant. The destination must be a word address.

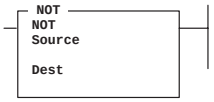
Application Note: When entering constants, you can use the ampersand (&) operator to change the radix of your entry. For example, instead of entering –1 as a constant, you could enter &B1111111111111111 or &HFFFF.

Updates to Arithmetic Status Bits

With this Bit:		The Controller:
S:0/0	Carry (C)	always resets.
S:0/1	Overflow (V)	always resets.
S:0/2	Zero (Z)	sets if result is zero; otherwise resets
S:0/3	Sign (S)	sets if result is negative (most significant bit is set); otherwise resets.

Not (NOT)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

This instruction performs a bit-by-bit logical NOT. The operation is performed using the value at source A. The result (one's complement of A) is stored in the destination.

Truth Table

Dest = NOT A	
A	**Dest**
0	1
1	0

The source and destination must be word addresses.

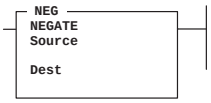
Application Note: When entering constants, you can use the ampersand (&) operator to change the radix of your entry. For example, instead of entering –1 as a constant, you could enter &B1111111111111111 or &HFFFF.

Updates to Arithmetic Status Bits

With this Bit:		The Controller:
S:0/0	Carry (C)	always resets.
S:0/1	Overflow (V)	always resets.
S:0/2	Zero (Z)	sets if result is zero; otherwise resets.
S:0/3	Sign (S)	sets if result is negative (most significant bit is set); otherwise resets.

Negate (NEG)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Output Instruction

Use the NEG instruction to change the sign of the source and then place it in the destination. The destination contains the two's complement of the source. For example, if the source is 5, the destination would be –5.

The source and destination must be word addresses.

Updates to Arithmetic Status Bits

With this Bit:		The Controller:
S:0/0	Carry (C)	clears if 0 or overflow, otherwise sets.
S:0/1	Overflow (V)	sets if overflow, otherwise reset. Overflow occurs only if –32,768 is the source. On overflow, the minor error flag is also set. The value 32,767 is placed in the destination. If S:2/14 is set, then the unsigned, truncated overflow remains in the destination. For floating point destinations, the overflow result remains in the destination.
S:0/2	Zero (Z)	sets if result is zero; otherwise resets.
S:0/3	Sign (S)	sets if result is negative; otherwise resets.

FIFO and LIFO Instructions Overview

FIFO instructions load words into a file and unload them in the same order as they were loaded. The first word in is the first word out.

LIFO instructions load words into a file and unload them in the opposite order as they were loaded. The last word in is the first word out.

Entering Parameters

Enter the following parameters when programming these instructions:

- **Source** is a word address or constant (–32,768 to 32,767) that becomes the next value in the stack.
- **Destination** is a word address that stores the value that exits from the stack.

This Instruction:	Unloads the Value from:
FIFO's FFU	First word
LIFO's LFU	The last word entered

- **FIFO/LIFO** is the address of the stack. It must be an indexed word address in the bit, input, output, or integer file. Use the same FIFO address for the associated FFL and FFU instructions; use the same LIFO address for the associated LFL and LFU instructions.
- **Length** specifies the maximum number of words in the stack. For SLC processors this is 128 words and 105 words for MicroLogix 1000 controllers. Address the length value by mnemonic (LEN).
- **Position** is the next available location where the instruction loads data into the stack. This value changes after each load or unload operation. Address the position value by mnemonic (POS).
- **Control** is a control file address. The status bits, the stack length, and the position value are stored in this element. Do not use the control file address for any other instruction.

Status bits of the control structure are addressed by mnemonic. These include:

- **Empty Bit EM** (bit 12) is set by the processor to indicate the stack is empty.
- **Done Bit DN** (bit 13) is set by the processor to indicate the stack is full. This inhibits loading the stack.
- **FFU/LFU Enable Bit EU** (bit 14) is set on a false-to-true transition of the FFU/LFU rung and is reset on a true-to-false transition.
- **FFL/LFL Enable Bit EN** (bit 15) is set on a false-to-true transition of the FFL/LFL rung and is reset on a true-to-false transition.

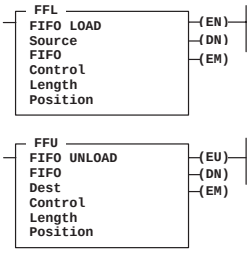
Effects on Index Register S:24

The value present in S:24 is overwritten with the position value when a false-to–true transition of the FFL/FFU or LFL/LFU rung occurs. For the FFL/LFL, the position value determined at instruction entry is placed in S:24. For the FFU/LFU, the position value determined at instruction exit is placed in S:24.

When the DN bit is set, a false-to–true transition of the FFL/LFL rung does not change the position value or the index register value. When the EM bit is set, a false-to–true transition of the FFU/LFU rung does not change the position value or the index register value.

FIFO Load (FFL) and FIFO Unload (FFU)

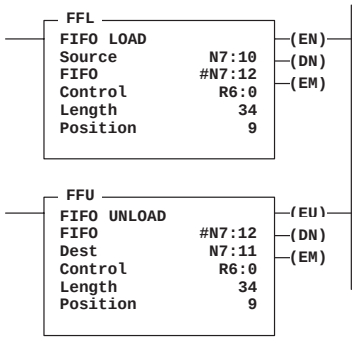
ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓



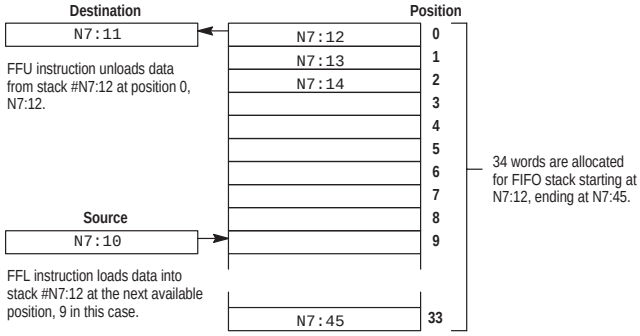
Output Instructions

FFL and FFU instructions are used in pairs. The FFL instruction loads words into a user-created file called a FIFO stack. The FFU instruction unloads words from the FIFO stack, in the same order as they were entered.

Instruction parameters have been programmed in the FFL–FFU instruction pair shown below.



FFL–FFU Instruction Pair



Loading and Unloading of Stack #N7:12

FFL Instruction Operation: When rung conditions change from false-to-true, the FFL enable bit (EN) is set. This loads the contents of the source, N7:10, into the stack element indicated by the position number, 9. The position value then increments.

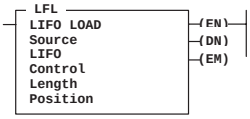
The FFL instruction loads an element at each false-to-true transition of the rung, until the stack is filled (34 elements). The processor then sets the done bit (DN), inhibiting further loading.

FFU Instruction Operation: When rung conditions change from false-to-true, the FFU enable bit (EU) is set. This unloads the contents of the element at stack position 0 into the destination, N7:11. All data in the stack is shifted one element toward position zero, and the highest numbered element is zeroed. The position value then decrements.

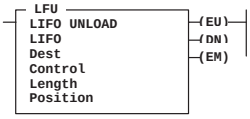
The FFU instruction unloads an element at each false–to-true transition of the rung, until the stack is empty. The processor then sets the empty bit (EM).

LIFO Load (LFL) and LIFO Unload (LFU)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓

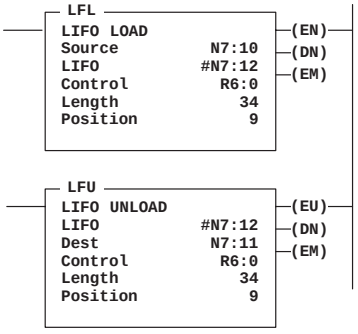


LFL and LFU instructions are used in pairs. The LFL instruction loads words into a user–created file called a LIFO stack. The LFU instruction unloads words from the LIFO stack in the opposite order as they were entered.

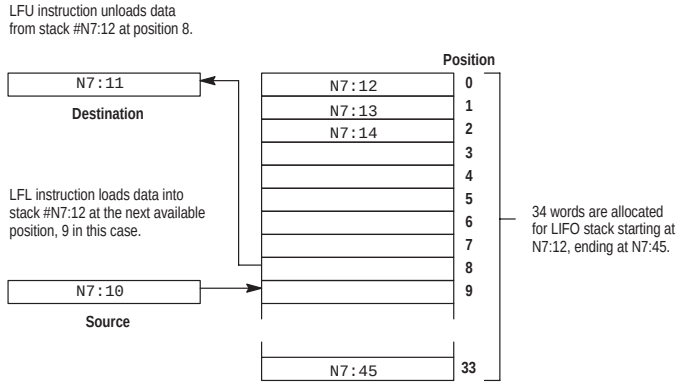


Instruction parameters have been programmed in the LFL – LFU instruction pair shown below.

Output Instructions



LFL–LFU Instruction Pair



Loading and Unloading of Stack #N7:12

LFL Instruction Operation: When rung conditions change from false-to-true, the LFL enable bit (EN) is set. This loads the contents of the source, N7:10, into the stack element indicated by the position number, 9. The position value then increments.

The LFL instruction loads an element at each false-to-true transition of the rung, until the stack is filled (34 elements). The processor then sets the done bit (DN), inhibiting further loading.

LFU Instruction Operation: When rung conditions change from false-to-true, the LFU enable bit (EU) is set. This unloads data from the last element loaded into the stack (at the position value minus 1), placing it in the destination, N7:11. The position value then decrements.

The LFU instruction unloads one element at each false-to-true transition of the rung, until the stack is empty. The processor then sets the empty bit (EM).

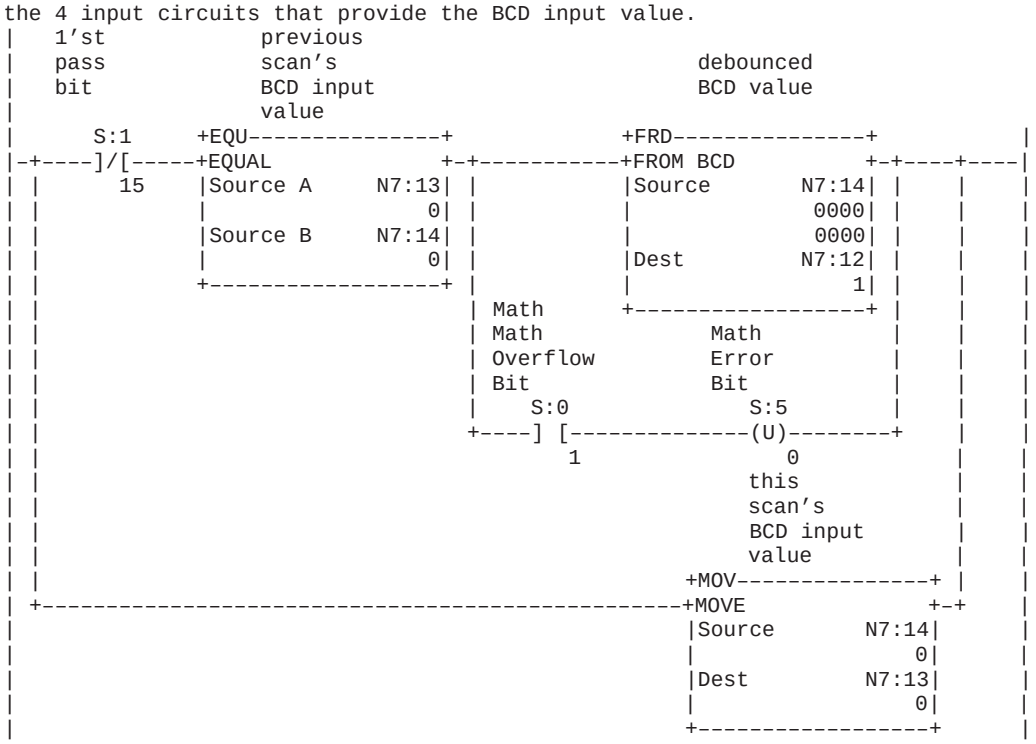
Data Handling Instructions in the Paper Drilling Machine Application Example

This section provides ladder rungs to demonstrate the use of data handling instructions. The rungs are part of the paper drilling machine application example described in appendix H. You will be adding to the subroutine in file 7 that was started in chapter 1.

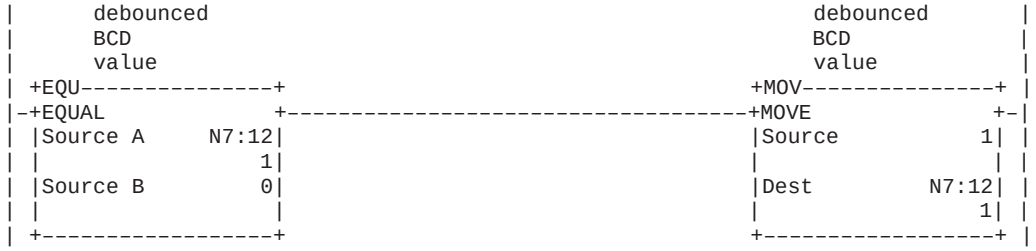
Adding File 7

```
Rung 7:3
This rung moves the single digit BCD thumbwheel value into an internal Integer
register. This is done to properly align the four BCD input signals prior to
executing the BCD to Integer instruction (FRD). The thumbwheel is used to allow
the operator to enter the thickness of the paper that is to be drilled. The
thickness is entered in 1/4″ increments. This provides a range of 1/4″ to 2.25″
|                                                    BCD bit 0 |FRD bit 0    |
|                                                      I:1.0      N7:14      |
|----------------------------------------------------+----] [--------( )-----+-|
|                                                    |      11          0    | |
|                                                    | BCD bit 1 |FRD bit 1  | |
|                                                    |   I:1.0      N7:14    | |
|                                                    +----] [--------( )-----+ |
|                                                    |      12          1    | |
|                                                    | BCD bit 2 |FRD bit 2  | |
|                                                    |   I:1.0      N7:14    | |
|                                                    +----] [--------( )-----+ |
|                                                    |      13          2    | |
|                                                    | BCD bit 3 |FRD bit 3  | |
|                                                    |   I:1.0      N7:14    | |
|                                                    +----] [--------( )-----+ |
|                                                           14          3      |
```

Rung 7:4
This rung converts the BCD thumbwheel value from BCD to Integer. This is done because the processor operates upon Integer values. This rung also ”debounces” the thumbwheel to ensure that the conversion only occurs on valid BCD values. Note that invalid BCD values can occur while the operator is changing the BCD thumbwheel. This is due to input filter propagation delay differences between the 4 input circuits that provide the BCD input value.



Rung 7:5
This rung ensures that the operator cannot select a paper thickness of 0. If this were allowed, the drill bit life calculation could be defeated, resulting in poor quality holes due to a dull drill bit. Therefore, the minimum paper thickness that is used to calculate drill bit wear is 1/4”.



5 Program Flow Instructions

This chapter contains general information about the program flow instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the program flow control instructions in use.

Program Flow Control Instructions

Instruction		Purpose	Page
Mnemonic	Name		
JMP and LBL	Jump to Label and Label	Jump forward or backward to the specified label instruction.	**5–2**
JSR, SBR, and RET	Jump to Subroutine, Subroutine, and Return from Subroutine	Jump to a designated subroutine and return.	**5–3**
MCR	Master Control Reset	Turn off all non–retentive outputs in a section of ladder program.	**5–6**
TND	Temporary End	Mark a temporary end that halts program execution.	**5–7**
SUS	Suspend	Identifies specific conditions for program debugging and system troubleshooting.	**5–8**
IIM	Immediate Input with Mask	Program an Immediate Input with Mask.	**5–8**
IOM	Immediate Output with Mask	Program an Immediate Output with Mask.	**5–9**
REF	Refresh	Interrupt the program scan to execute the I/O scan and service communications.	**5–10**

About the Program Flow Control Instructions

Use these instructions to control the sequence in which your program is executed.

Control instructions allow you to change the order in which the processor scans a ladder program. Typically, these instructions are used to minimize scan time, create a more efficient program, and troubleshoot a ladder program.

Jump (JMP) and Label (LBL)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓

—(JMP)—

——] LBL [——

Use these instructions in pairs to skip portions of the ladder program.

If the Rung Containing the Jump Instruction is:	Then the Program:
True	Skips from the rung containing the JMP instruction to the rung containing the designated LBL instruction and continues executing. You can jump forward or backward.
False	Does not execute the JMP instruction.

Jumping forward to a label saves program scan time by omitting a program segment until needed. Jumping backward lets the controller execute program segments repeatedly.

Note *Be careful not to jump backwards an excessive number of times. The watchdog timer could time out and fault the controller. Use a counter, timer, or the "program scan" register (system status register, word S:3, bits 0–7) to limit the amount of time you spend looping inside of JMP/LBL instructions.*

Entering Parameters

Enter a decimal label number from 0 to 999. You can place up to:

- 256 labels for SLC processors in each subroutine file
- 1,000 labels for MicroLogix 1000 controllers in each subroutine file

Using JMP

The JMP instruction causes the controller to skip rungs. You can jump to the same label from one or more JMP instruction.

Using LBL

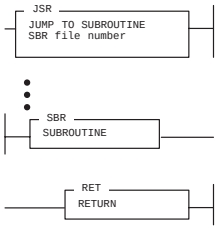
This input instruction is the target of JMP instructions having the same label number. You must program this instruction as the first instruction of a rung. This instruction has no control bits.

You can program multiple jumps to the same label by assigning the same label number to multiple JMP instructions. However, label numbers must be unique.

Note *Do not jump (JMP) into an MCR zone. Instructions that are programmed within the MCR zone starting at the LBL instruction and ending at the 'END MCR' instruction are always evaluated as though the MCR zone is true, regardless of the true state of the "Start MCR" instruction.*

Jump to Subroutine (JSR), Subroutine (SBR), and Return (RET)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓



The JSR, SBR, and RET instructions are used to direct the controller to execute a separate subroutine file within the ladder program and return to the instruction following the JSR instruction.

Note *If you use the SBR instruction, the SBR instruction must be the first instruction on the first rung in the program file that contains the subroutine.*

Use a subroutine to store recurring sections of program logic that must be executed from several points within your application program. A subroutine saves memory because you program it only once.

Update critical I/O within subroutines using immediate input and/or output instructions (IIM, IOM), especially if your application calls for nested or relatively long subroutines. Otherwise, the controller does not update I/O until it reaches the end of the main program (after executing all subroutines).

Outputs controlled within a subroutine remain in their last state until the subroutine is executed again.

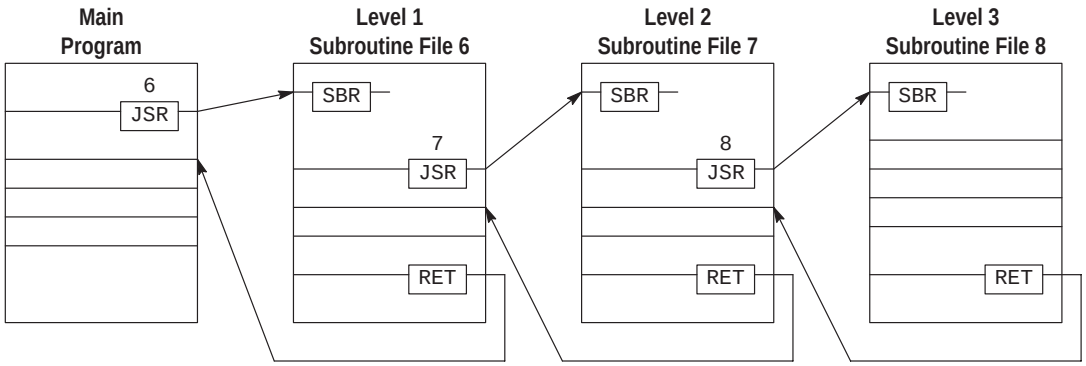
Nesting Subroutine Files

Nesting subroutines allows you to direct program flow from the main program to a subroutine and then on to another subroutine. The following rules apply when nesting subroutines:

You can nest up to eight levels of subroutines. If you are using an STI subroutine, HSC interrupt subroutine, or user fault routine, you can nest subroutines up to three levels from each subroutine.

- With Fixed and SLC 5/01 processors, you can nest subroutines up to four levels.
- With SLC 5/02 and higher processors and MicroLogix 1000 controllers, you can nest subroutines up to eight levels. If you are using an STI subroutine, I/O event–driven interrupt subroutine, user fault routine, or HSC interrupt subroutine, you can nest subroutines up to three levels from each subroutine.

The following figure illustrates how subroutines may be nested.



Example of Nesting Subroutines to Level 3

An error occurs if more than the allowable levels of subroutines are called (subroutine stack overflow) or if more returns are executed than there are call levels (subroutine stack underflow).

Using JSR

When the JSR instruction is executed, the controller jumps to the subroutine instruction (SBR) at the beginning of the target subroutine file and resumes execution at that point. You cannot jump into any part of a subroutine except the first instruction in that file.

You must program each subroutine in its own program file by assigning a unique file number:

- 3–255 for SLC processors
- 4–15 for MicroLogix 1000 controllers

Fixed and SLC 5/01 specific – The JSR instruction should not be programmed in nested output branches. A compiler error will occur if a rung containing multiple outputs with conditional logic and a JSR instruction is encountered.

Using SBR

The target subroutine is identified by the file number that you entered in the JSR instruction. This instruction serves as a label or identifier for a program file as a regular subroutine file.

This instruction has no control bits. It is always evaluated as true. The instruction must be programmed as the first instruction of the first rung of a subroutine. Use of this instruction is optional; however, we recommend using it for clarity.

Using RET

This output instruction marks the end of subroutine execution or the end of the subroutine file. It causes the controller to resume execution at the instruction following the JSR instruction. If a sequence of nested subroutines is involved, the instruction causes the processor to return program execution to the previous subroutine.

The rung containing the RET instruction may be conditional if this rung precedes the end of the subroutine. In this way, the controller omits the balance of a subroutine only if its rung condition is true.

Without an RET instruction, the END instruction (always present in the subroutine) automatically returns program execution to the instruction following the JSR instruction in your calling ladder file.

Note *The RET instruction terminates execution of the DII subroutine (SLC 5/03 and higher processors), STI subroutine, I/O event-driven interrupt subroutine, and the user error handler when a SLC 5/02 or higher processor is used.*

Master Control Reset (MCR)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓

—(MCR)—

Use MCR instructions in pairs to create program zones that turn off all the non-retentive outputs in the zone. Rungs within the MCR zone are still scanned, but scan time is reduced due to the false state of non-retentive outputs.

If the MCR Rung that Starts the Zone is:	Then the Controller:
True	Executes the rungs in the MCR zone based on each rung's individual input condition (as if the zone did not exist).
False	Resets all non–retentive output instructions in the MCR zone regardless of each rung's individual input conditions.

MCR zones let you enable or inhibit segments of your program, such as for recipe applications.

When you program MCR instructions, note that:

- You must end the zone with an unconditional MCR instruction.
- You cannot nest one MCR zone within another.
- Do not jump into an MCR zone. If the zone is false, jumping into it activates the zone.
- Always place the MCR instruction as the last instruction in a rung.

Note

The MCR instruction is not a substitute for a hard–wired master control relay that provides emergency stop capability. You still must install a hard–wired master control relay to provide emergency I/O power shutdown.

If you start instructions such as timers or counters in an MCR zone, instruction operation ceases when the zone is disabled. Re–program critical operations outside the zone if necessary.

SLC Processor Operation

Do not jump (JMP) into an MCR zone. Instructions that are programmed within the MCR zone starting at the LBL instruction and ending at the 'END MCR' instruction are always evaluated as though the MCR zone is true, regardless of the true state of the "Start MCR" instruction. If the zone is false, jumping into it activates the zone from the LBL to the end of the zone.

If you start instructions such as timers or counters in an MCR zone, instruction operation ceases when the zone is disabled. Re-program critical operations outside the zone if necessary.

The TOF timer activates when placed inside of a false MCR zone.

The MCR instruction is not a substitute for a hard-wired master control relay. We recommend that your programmable controller system include a hard-wired master control relay and emergency stop switches to provide I/O power shut down. Emergency stop switches can be monitored but should not be controlled by the ladder program. Wire these devices as described in the installation manual.

SLC 5/03 and and higher processors – When online and an unmatched MCR instruction exists in your program, the END instruction acts as the second unconditional MCR instruction and all of the rungs following the first MCR instruction execute via the current MCR instruction state.

You can save the program while online if unattended MCR instructions exist. However, if you are offline and unattended MCR instructions exist, an error will occur.

Temporary End (TND)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓

—(TND)—

Output Instruction

This instruction, when its rung is true, stops the processor from scanning the rest of the program file, updates the I/O, and resumes scanning at rung 0 of the main program (file 2). If this instruction's rung is false, the processor continues the scan until the next TND instruction or the END statement. Use this instruction to progressively debug a program, or conditionally omit the balance of your current program file or subroutines.

Note

If you use this instruction inside a nested subroutine, execution of all nested subroutines is terminated.

MicroLogix 1000 controllers – Do not execute this instruction from the user error fault routine (file 3), high–speed counter interrupt routine (file 4), or selectable timed interrupt routine (file 5) because a fault will occur.

Suspend (SUS)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓

Output Instruction

When this instruction is executed, it causes the processor to enter the Suspend Idle mode and stores the Suspend ID in word 7 (S:7) of the status file. All outputs are de-energized.

Use this instruction to trap and identify specific conditions for program debugging and system troubleshooting.

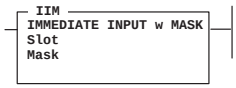
Entering Parameters

Enter a suspend ID number from –32,768 to +32,767 when you program the instruction.

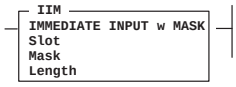
When the SUS instruction is executed, the programmed ID as well as the program file ID from which the SUS instruction executed is placed in the system status file.

Immediate Input with Mask (IIM)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



Input Instruction
Fixed andSLC 5/01 processors
and MicroLogix 1000 controllers



Input Instruction
SLC 5/03 and higher Processors

This instruction allows you to update data prior to the normal input scan. When the IIM instruction is enabled, the program scan is interrupted. Data from a specified I/O slot is transferred through a mask to the input data file, making the data available to instructions following the IIM instruction in the ladder program.

For the mask, a 1 in an input's bit position passes data from the source to the destination. A 0 inhibits data from passing from the source to the destination.

Entering Parameters

Slot – Specify the input slot number and the word number pertaining to the slot. Word 0 of a slot need not be specified. Fixed and SLC 5/01 processors can have up to 8 words associated with the slot. The SLC 5/02 and higher processors can have up to 32 words associated with the slot (0–31).

For all MicroLogix 1000 controllers specify I1:0.0. For 16 I/O controllers, I1:0/0–9 are valid and I1:0/10–15 are considered unused inputs. (They do not physically exist.) For 32 I/O controllers, I1:0/0–15 and I1:1/0–3 are valid. Specify I1:1 if you want to immediately update the last four input bits.

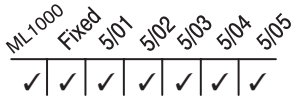
Example

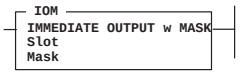
I:2	Inputs of slot 2, word 0
I:2.1	Inputs of slot 2, word 1
I:1	Inputs of slot 1, word 0

Mask – Specify a hexadecimal constant or register address.

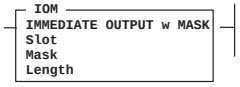
Length – For SLC 5/03 and higher processors, this parameter is used to transfer more than one word per slot.

Immediate Output with Mask (IOM)





Output Instruction
Fixed and SLC 5/01 processors
and MicroLogix 1000 controllers



Output Instruction
SLC 5/03 and higher Processors

This instruction allows you to update the outputs prior to the normal output scan. When the IOM instruction is enabled, the program scan is interrupted to transfer data to a specified I/O slot through a mask. The program scan then resumes.

For the mask, a 1 in the output bit position passes data from the source to the destination. A 0 inhibits the data from passing from the source to the destination.

Entering Parameters

Slot – Specify the slot number and the word number pertaining to the slot. Word 0 of a slot need not be specified. Fixed and SLC 5/01 processors can have up to 8 words associated with the slot. The SLC 5/02 and higher processors can have up to 32 words associated with the slot (0–31).

For all MicroLogix 1000 controllers specify O0:0.0. For 16 I/O controllers, O0:0/0–5 are valid and O0:0/6–15 are considered unused outputs. (They do not physically exist.) For 32 I/O controllers, O0:0/0–11 are valid and O0:0/12–15 are considered unused outputs.

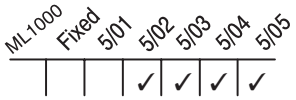
Example

O:2	Outputs of slot 2, word 0
O:1	Outputs of slot 1, word 0
O:2.1	Outputs of slot 2, word 1

Mask – Specify a hexadecimal constant or register address.

Length – For SLC 5/03 and higher processors, this parameter is used to transfer more than one word per slot.

I/O Refresh (REF)



Using an SLC 5/02 Processor

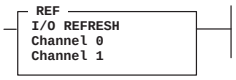
—(REF)—
Output Instruction

The REF instruction has no programming parameters. When it is evaluated as true, the program scan is interrupted to execute the I/O scan and service communication portions of the operating cycle (write outputs, service comms, read inputs). The scan then resumes at the instruction following the REF instruction.

You are not allowed to place a REF instruction in a DII subroutine, STI subroutine, I/O subroutine, or user fault subroutine.

The watchdog and scan timers are reset when executing the REF instruction. You must insure that an REF instruction is not placed inside a non-terminating program loop. Do not place an REF instruction inside a program loop unless the program is thoroughly analyzed.

Using SLC 5/03 and Higher Processors



Output Instruction

Operation of the REF instruction in the SLC 5/03 and higher processors is the same as the SLC 5/02 processor. However, when using the SLC 5/03 and higher processors, you can also select a specific communication channel to be serviced.

- SLC 5/03 processor
  - channel 0 is RS-232/DF1 Full-Duplex or Half-Duplex (master or slave), DH-485, or ASCII
  - channel 1 is DH-485
- SLC 5/04 processor
  - channel 0 is RS-232/DF1 Full-Duplex or Half-Duplex (master or slave), DH-485, or ASCII
  - channel 1 is DH+
- SLC 5/05 processor
  - channel 0 is RS-232/DF1 Full-Duplex or Half-Duplex (master or slave), DH-485, or ASCII
  - channel 1 is Ethernet

Program Flow Control Instructions in the Paper Drilling Machine Application Example

This section provides ladder rungs to demonstrate the use of program flow control instructions. The rungs are part of the paper drilling machine application example described in appendix H. You will be adding to the main program in file 2. The new rungs are needed to call the other subroutines containing the logic necessary to run the machine.

Adding File 2

```
Rung 2:3
This rung calls the drill sequence subroutine. This subroutine manages the
operation of a drilling sequence and restarts the conveyer upon completion of
the drilling sequence
|                                                          +JSR---------------+ |
|----------------------------------------------------------+JUMP TO SUBROUTINE+-|
|                                                          |SBR file number  6| |
|                                                          +------------------+ |
Rung 2:4
This rung calls the subroutine that tracks the amount of wear on the current
drill bit.
|                                                          +JSR---------------+ |
|----------------------------------------------------------+JUMP TO SUBROUTINE+-|
|                                                          |SBR file number  7| |
|                                                          +------------------+ |
Rung 2:5
There is some initialization logic in the DII subroutine (file 4) that must be
executed prior to the first DII interrupt.  So this rung allows the DII to be
initialized by jumping to the DII subroutine when the processor enters the RUN
mode.
| 1st                                                                          |
| Pass                                                                         |
|    S:1                                                   +JSR---------------+ |
|----] [---------------------------------------------------+JUMP TO SUBROUTINE+-|
|      15                                                  |SBR file number  4| |
|                                                          +------------------+ |
```

6 Application Specific Instructions

This chapter contains general information about the application specific instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the application specific instructions in use.

Application Specific Instructions

Instruction		Purpose	Page
Mnemonic	**Name**		
BSL and BSR	Bit Shift Left and Bit Shift Right	Loads a bit of data into a bit array, shifts the pattern of data through the array, and unloads the last bit of data in the array. The BSL shifts data to the left and the BSR shifts data to the right.	**6–4**
SQO and SQC	Sequencer Output and Sequencer Compare	Controls sequential machine operations by transferring 16-bit data through a mask to image addresses.	**6–7**
SQL	Sequencer Load	Captures referenced conditions by manually stepping the machine through its operating sequences.	**6–13**

About the Application Specific Instructions

These instructions simplify your ladder program by allowing you to use a single instruction or pair of instructions to perform common complex operations.

In this chapter you will find a general overview preceding groups of instructions. Before you learn about the instructions in each of these groups, we suggest that you read the overview. This chapter contains the following overviews:

- Bit Shift Instructions Overview
- Sequencer Instructions Overview

Bit Shift Instructions Overview

The following general information applies to bit shift instructions.

Entering Parameters

Enter the following parameters when programming these instructions:

- **File** is the address of the bit array you want to manipulate. You must use the file indicator (#) in the bit array address.
- **Control** is the control element that stores the status byte of the instruction and the size of the array (in number of bits). Note that the control address should not be used for any other instruction.

The control element is shown below.

	15 13	11 10	00
Word 0	EN DN	ER UL	Not used
Word 1	Size of bit array (number of bits)		
Word 2	Reserved		

Status bits of the control element may be addressed by mnemonic. They include:

- **Unload Bit UL** (bit 10) stores the status of the bit exited from the array each time the instruction is enabled.
- **Error Bit ER** (bit 11), when set, indicates the instruction detected an error such as entering a negative number for the length or position. Avoid using the output bit when this bit is set.
- **Done Bit DN** (bit 13), when set, indicates the bit array has shifted one position.
- **Enable Bit EN** (bit 15) is set on a false-to-true transition of the rung and indicates the instruction is enabled.

When the register shifts and input conditions go false, the enable, done, and error bits are reset.

- **Bit Address** is the address of the source bit that the instruction inserts in the first (lowest) bit position (BSL) or the last (highest) bit position (BSR).
- **Length** (size of bit array) is the number of bits in the bit array, up to 2048 bits. A length value of 0 causes the input bit to be transferred to the UL bit.

  – For SLC processors, the length is 2048.

  – For MicroLogix 1000 controllers, this length is 1680.

  A length value that points past the end of the programmed file causes a runtime major error to occur.

Note *If you alter a length value with your ladder program, make certain that the altered value is valid.*

The instruction invalidates all bits beyond the last bit in the array (as defined by the length) up to the next word boundary.

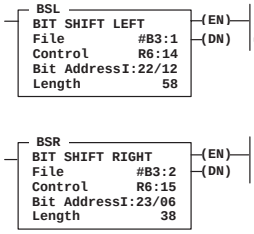
Note *If a STring element address is used for the file parameter, the maximum length for SLC 5/03 and higher processors is 672 bits. Additionally, STring element boundaries cannot be crossed.*

Effects on Index Register S:24

The shift operation clears the index register S:24 to zero.

Bit Shift Left (BSL)
Bit Shift Right (BSR)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



BSL and BSR are output instructions that load data into a bit array one bit at a time. The data is shifted through the array, then unloaded one bit at a time.

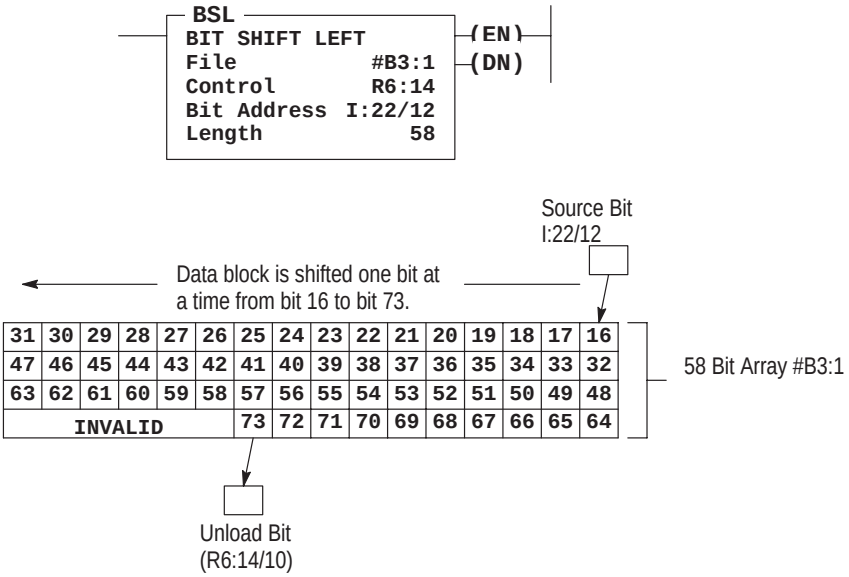
Output Instructions

Using BSL

When the rung goes from false-to-true, the processor sets the enable bit (EN bit 15) and the data block is shifted to the left (to a higher bit number) one bit position. The specified bit at the bit address is shifted into the first bit position. The last bit is shifted out of the array and stored in the unload bit (UL bit 10). The shift is completed immediately.

For wraparound operation, set the position of the bit address to the last bit of the array or to the UL bit, whichever applies.

The figure below illustrates how the Bit Shift Left instruction works.



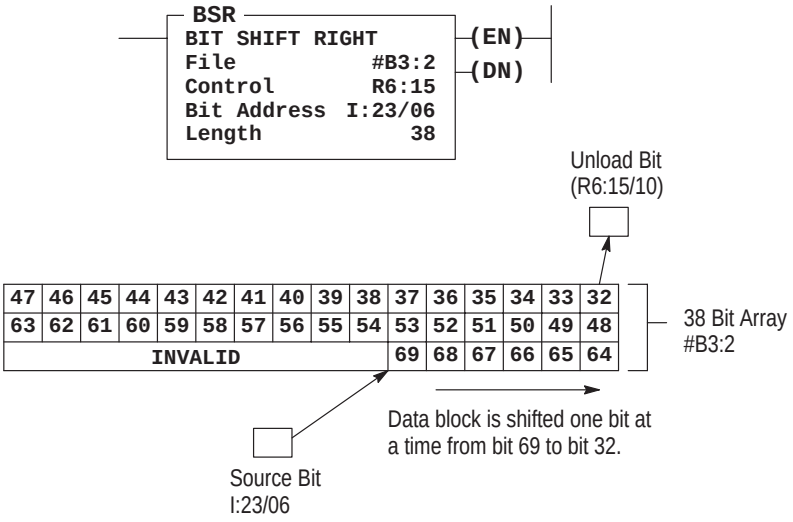
If you wish to shift more than one bit per scan, you must create a loop in your application using the JMP, LBL, and CTU instructions.

Using BSR

When the rung goes from false-to-true, the enable bit (EN bit 15) is set and the data block is shifted to the right (to a lower bit number) one bit position. The specified bit at the bit address is shifted into the last bit position. The first bit is shifted out of the array and stored in the unload bit (UL bit 10) in the status byte of the control element. The shift is completed immediately.

For wraparound operation, set the position of the bit address to the first bit of the array or to the UL bit, whichever applies.

The figure below illustrates how the Bit Shift Right instruction works.



If you wish to shift more than one bit per scan, you must create a loop in your application using the JMP, LBL, and CTU instructions.

Sequencer Instructions Overview

The following general information applies to sequencer instructions.

Effects on Index Register S:24

The value present in the index register S:24 is overwritten when the sequencer instruction is true. The index register value will equal the position value of the instruction.

Applications Requiring More than 16-Bits

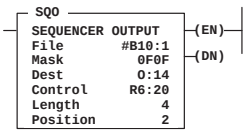
When your application requires more than 16-bits, use parallel multiple sequencer instructions.

Note *Refer to appendix H for application examples using the sequencer instructions.*

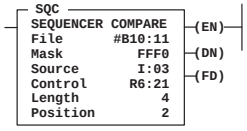
Note *If a STring element address is used for the file parameter, the maximum length for SLC 5/03 and higher processors is 41 words. Additionally, STring element boundaries cannot be crossed.*

Sequencer Output (SQO) Sequencer Compare (SQC)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓	✓	✓	✓	✓	✓



These instructions transfer 16-bit data to word addresses for the control of sequential machine operations.



Output Instructions

Entering Parameters

Enter the following parameters when programming these instructions:

- **File** is the address of the sequencer file. You must use the file indicator (#) for this address.

  Sequencer file data is used as follows:

Instruction	Sequencer File Stores
SQO	Data for controlling outputs
SQC	Reference data for monitoring inputs

- **Mask** (SQO, SQC) is a hexadecimal code or the address of the mask word or file through which the instruction moves data. Set mask bits to pass data and reset mask bits to mask data. Use a mask word or file if you want to change the mask according to application requirements.

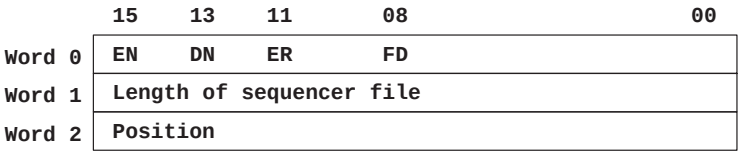
  If the mask is a file, its length will be equal to the length of the sequencer file. The two files track automatically.

- **Source** is the address of the input word or file for a SQC from which the instruction obtains data for comparison to its sequencer file.
- **Destination** is the address of the output word or file for a SQO to which the instruction moves data from its sequencer file.

Note

You can address the mask, source, or destination of a sequencer instruction as a word or file. If you address it as a file (using file indicator #), the instruction automatically steps through the source, mask, or destination file.

- **Control** (SQO, SQC) is the control structure that stores the status byte of the instruction, the length of the sequencer file, and the instantaneous position in the file. You should not use the control address for any other instruction.



Status bits of the control structure include:

- **Found Bit FD** (bit 08) – SQC only. When the status of all non-masked bits in the source address match those of the corresponding reference word, the FD bit is set. This bit is assessed each time the SQC instruction is evaluated while the rung is true.
- **Error Bit ER** (bit 11) is set when the processor detects a negative position value, or a negative or zero length value. This results in a major error if not cleared before the END or TND instruction is executed.
- **Done Bit DN** (bit 13) is set by the SQO or SQC instruction after it has operated on the last word in the sequencer file. It is reset on the next false-to-true rung transition after the rung goes false.
- **Enable EN** (bit 15) is set by a false-to-true rung transition and indicates the SQO or SQC instruction is enabled.

- **Length** is the number of steps of the sequencer file starting at position 1. The maximum number you can enter is 255 words (104 words when using MicroLogix 1000 controllers). Position 0 is the startup position. The instruction resets (wraps) to position 1 at each cycle completion.

  The address assigned for a sequencer file is step zero. Sequencer instructions use length + 1 word of data table files for each file referenced in the instruction. This applies to the source, mask, and/or destination if addressed as files.

  A length value that points past the end of the programmed file causes a runtime major error to occur.

Note *If you alter a length value with your ladder program, make certain that the altered value is valid.*

- **Position** is the word location or step in the sequencer file from/to which the instruction moves data.

  A position value that points past the end of the programmed file causes a runtime major error to occur. *If you alter a position value with your ladder program, make certain that the altered value is valid.*

Application Note: You may use the reset (RES) instruction to reset a sequencer. All control bits (except FD) will be reset to zero. The Position will also be set to zero. Program the address of your control register in the RES (e.g.,R6:0).

Using SQO

This output instruction steps through the sequencer file whose bits have been set to control various output devices.

When the rung goes from false-to-true, the instruction increments to the next step (word) in the sequencer file. Data stored there is transferred through a mask to the destination address specified in the instruction. Current data is written to the corresponding destination word every scan that the rung remains true.

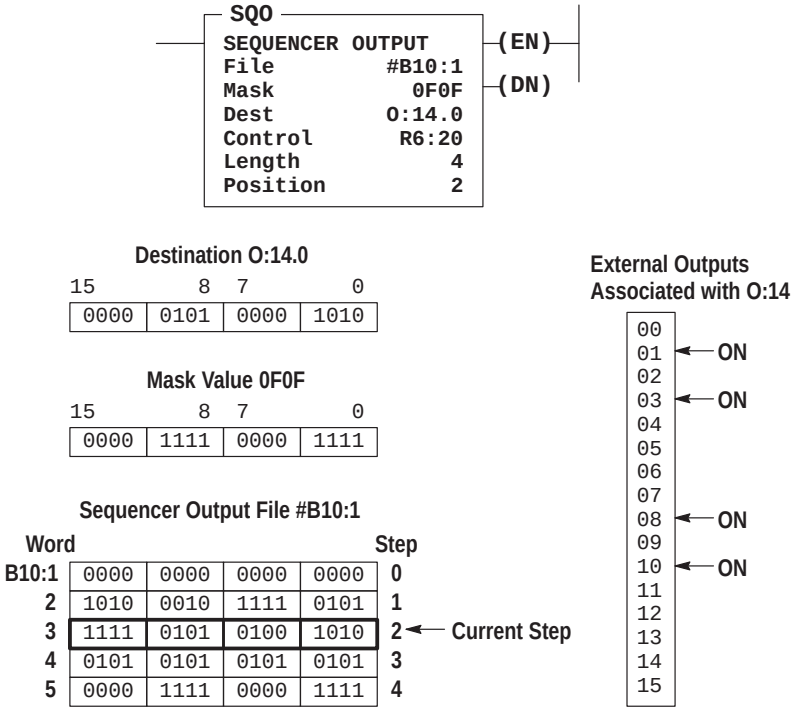
The done bit is set when the last word of the sequencer file is transferred. On the next false-to-true rung transition, the instruction resets the position to step one.

If the position is equal to zero at startup, when you switch the processor from the program mode to the run mode instruction operation depends on whether the rung is true or false on the first scan.

- If true, the instruction transfers the value in step zero.
- If false, the instruction waits for the first rung transition from false-to-true and transfers the value in step one.

The bits mask data when reset and pass data when set. The instruction will not change the value in the destination word unless you set mask bits. The mask can be fixed or variable. It will be variable if you enter an element address or a file address for changing the mask with each step.

The following figure indicates how the SQO instruction works.



Using SQC

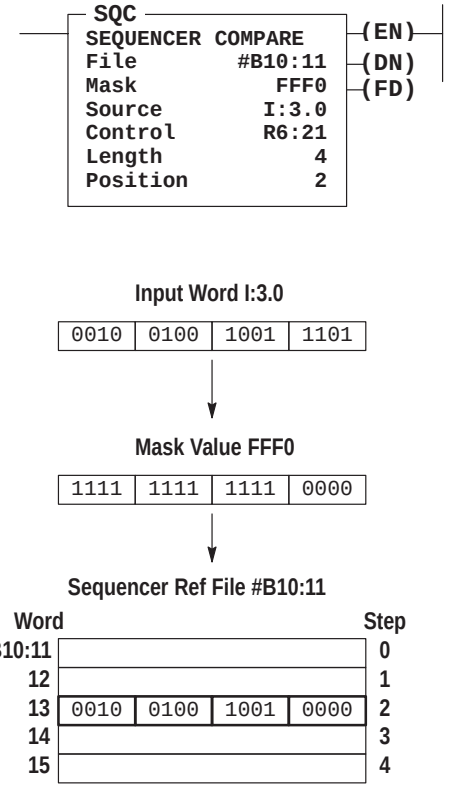
When the status of all non-masked bits in the source word match those of the corresponding reference word, the instruction sets the found bit (FD) in the control word. Otherwise, the found bit (FD) is cleared.

The bits mask data when reset and pass data when set.

The mask can be fixed or variable. If you enter a hexadecimal code, it is fixed. If you enter an element address or a file address for changing the mask with each step, it is variable.

When the rung goes from false-to-true, the instruction increments to the next step (word) in the sequencer file. Data stored there is transferred through a mask and compared against the source data for equality. If the source data equals the reference data, the FD bit is set in the SQC's control counter. Current data is compared against the source every scan that the rung evaluates as true.

Applications of the SQC instruction include machine diagnostics. The following figure explains how the SQC instruction works.

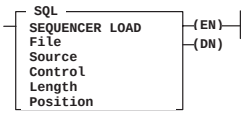


SQC FD bit is set when the instruction detects that an input word matches (through mask) its corresponding reference word.

The FD bit R6:21/FD is set in this example, since the input word matches the sequencer reference value using the mask value.

Sequencer Load (SQL)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓



Output Instruction

The SQL instruction stores 16-bit data into a sequencer load file at each step of sequencer operation. The source of this data can be an I/O or storage word address, a file address, or a constant.

Entering Parameters

Enter the following parameters when programming this instruction:

- **File** is the address of the sequencer file. You must use the file indicator (#) for this address.
- **Source** can be a word address, file address, or a constant (–32768 to 32767).

  If the source is a file address, the file length equals the length of the sequencer load file. The two files will step automatically, per the position value.

- **Length** is the number of steps of the sequencer load file (and also of the source if the source is a file address), starting at position 1. The maximum number you can enter is 255 words (104 words when using MicroLogix 1000 controllers). Position 0 is the startup position. The instruction resets (wraps) to position 1 at each cycle completion.

  The position address assigned for a sequencer file is step zero. Sequencer instructions use length plus one word of data for each file referenced in the instruction. This applies to the source if addressed as a file.

  A length value that points past the end of the programmed file causes a runtime major error to occur.

Note *If you alter a length value with your ladder program, make certain that the altered value is valid.*

- **Position** is the word location or step in the sequencer file to which data is moved.

  A position value that points past the end of the programmed file causes a runtime major error to occur.

Note *If you alter a position value with your ladder program, make certain that the altered value is valid.*

- **Control** is a control file address. The status bits, length value, and position value are stored in this element. Do not use the control file address for any other instruction.

  The control element is shown below:

	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
Word 0	EN		DN		ER											
Word 1	Length															
Word 2	Position															

  Status bits of the control structure include:

  – **Error Bit ER** (bit 11) is set when the processor detects a negative position value, or a negative or zero length value.

    For SLC processors, this results in a major error if not cleared before the END or TND instruction is executed.

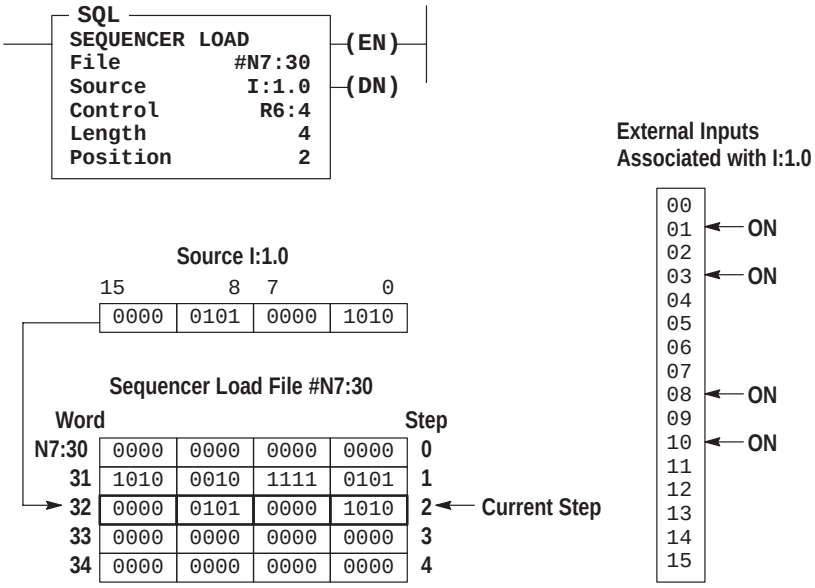
    For MicroLogix 1000 controllers, when the ER bit is set the minor error bit (S5:2) is also set. Both bits must be cleared.

  – **Done Bit DN** (bit 13) is set after the instruction has operated on the last word in the sequencer load file. It is reset on the next false-to-true rung transition after the rung goes false.

  – **Enable Bit EN** (bit 15) is set on a false-to-true transition of the SQL rung and reset on a true-to-false transition.

Operation

Instruction parameters have been programmed in the SQL instruction shown below. Input word I:1.0 is the source. Data in this word is loaded into integer file #N7:30 by the sequencer load instruction.



When rung conditions change from false-to-true, the SQL enable bit (EN) is set. The control element R6:4 increments to the next position in the sequencer file, and loads the contents of source I:1.0 into this location. The SQL instruction continues to load the current data into this location each scan that the rung remains true. When the rung goes false, the enable bit (EN) is reset.

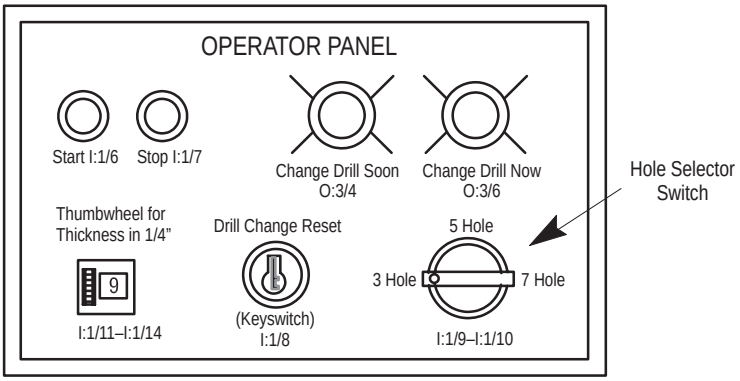
The instruction loads data into a new file element at each false–to–true transition of the rung. When step 4 is completed, the done bit (DN) is set. Operation cycles to position 1 at the next false–to–true transition of the rung after position 4.

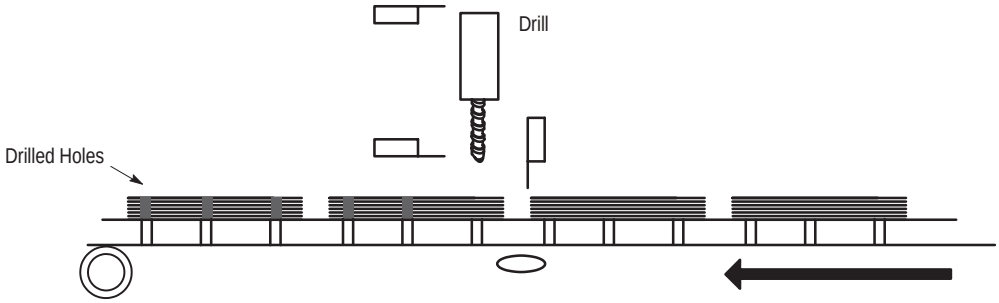
If the source were a file address such as #N7:40, files #N7:40 and #N7:30 would both have a length of 5 (0–4) and would track through the steps together per the position value.

Application Specific Instructions in the Paper Drilling Machine Application Example

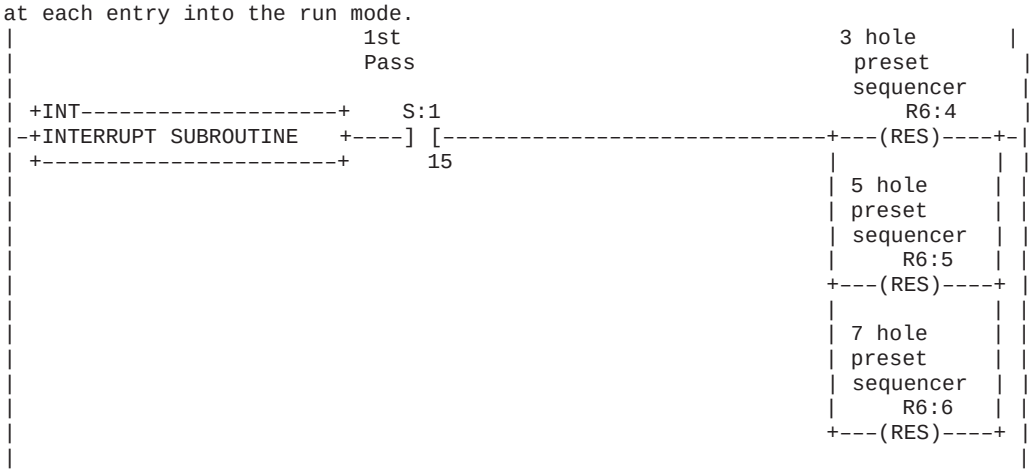
This section provides ladder rungs to demonstrate the use of application specific instructions. The rungs are part of the paper drilling machine application example described in appendix H. You will begin a subroutine in file 4.

This portion of the subroutine tells the conveyor where to stop to allow a hole to be drilled. Sequencer instructions are used to store conveyor stopping positions and to load the “next”stopping position into the DII Preset Word. (The Discrete Input Interrupt, DII, is used to count pulses coming from the encoder that is attached to the conveyor.) The stop positions are different for each hole pattern (3 hole, 5 hole, 7 hole), so separate sequencers are used to store and access each of the three hole patterns.

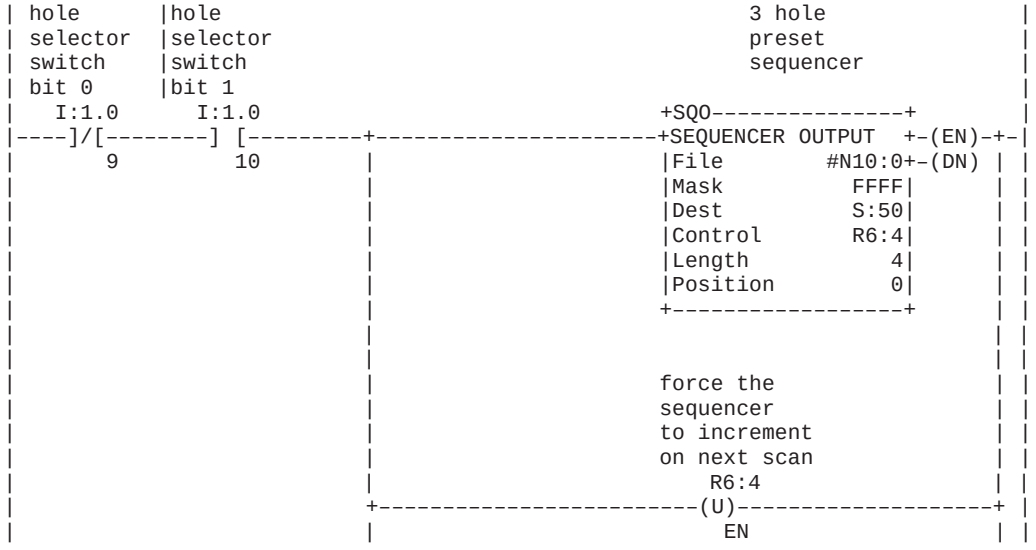




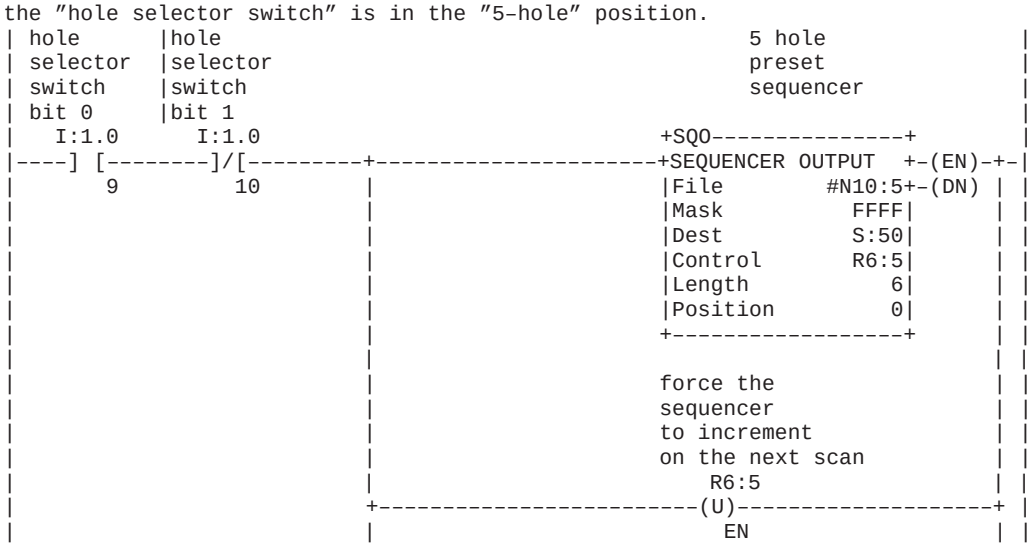
Rung 4:0
This rung resets the hole count sequencers each time the processor enters the RUN mode. This ensures that the first preset value is loaded into the DII preset at each entry into the run mode.



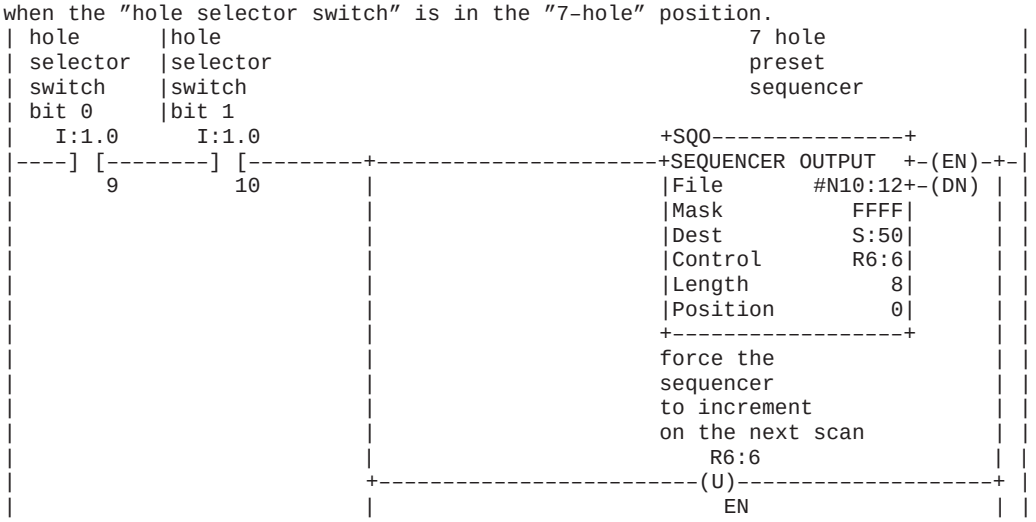
Rung 4:2
This rung keeps track of the hole number that is being drilled and loads the next correct DII preset based on the hole count. This rung is only active when the ”hole selector switch” is in the ”3–hole” position. The sequencer uses step 0 as a null step upon reset. It uses the last step as a ”go forever” in anticipation of the ”end of manual”. Moving a 0 into S:49 tells the DII to trigger an interrupt when the trailing edge of the current book is detected.



Rung 4:3
This rung is identical to the previous rung except that it is only active when the ”hole selector switch” is in the ”5–hole” position.



Rung 4:4
This rung is identical to the 2 previous rungs except that it is only active when the ”hole selector switch” is in the ”7–hole” position.



7 Using High-Speed Counter Instructions

This chapter contains general information about the high-speed counter instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

In addition, the last section contains an application example for a paper drilling machine that shows the high-speed counter instructions in use.

Note *The high-speed counter instruction is only supported by the SLC 500 Fixed and MicroLogix 1000 Controllers. The SLC modular controllers (SLC 5/01 thru SLC 5/05) do not have an on-board high-speed counter.*

High-Speed Counter Instructions

Mnemonic	Name	Purpose	Page
HSC	High-Speed Counter	Applies configuration to the high-speed counter hardware, updates the image accumulator, enables counting when the HSC is true, and disables counting when the HSC rung is false.	**7–6**
HSL	High-Speed Counter Load	Configures the low and high presets, the output patterns, and mask bit patterns.	**7–19**
RES	High-Speed Counter Reset	Writes a zero to the hardware accumulator and image accumulator.	**7–22**
RAC	High-Speed Counter Reset Accumulator	Writes the value specified to the hardware accumulator and image accumulator.	**7–23**
HSE **HSD**	High-Speed Counter Interrupt Enable High-Speed Counter Interrupt Disable	Enables or disables execution of the high-speed counter interrupt subroutine when a high preset, low preset, overflow, or underflow is reached.	**7–24**
OTE	Update High-Speed Counter Image Accumulator	Provides you with real-time access to the hardware accumulator value by updating the image accumulator.	**7–25**

About the High-Speed Counter Instructions

The high-speed counter instructions used in your ladder program configure, control, and monitor the controllers' hardware counter. The hardware counter's accumulator increments or decrements in response to external input signals. When the high-speed counter is enabled, data table counter C5:0 is used by the ladder program for monitoring the high-speed counter accumulator and status. The high-speed counter operates *independent* of the controller scan.

When using the high-speed counter, make sure you adjust your input filters accordingly.

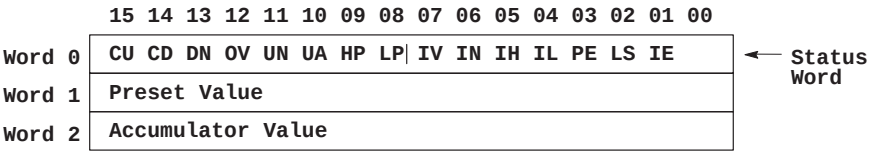
Before you learn about these instructions, read the overview that follows on the next page.

High-Speed Counter Instructions Overview

Use the high-speed counter instructions to detect and store narrow (fast) pulses, and to initiate other control operations based on preset values. These control operations include the automatic and immediate execution of the high-speed counter interrupt routine (file 4) and the immediate update of outputs based on a source and mask pattern you set.

Counter Data File Elements

The high-speed counter instructions reference counter C5:0. The HSC instruction is fixed at C5:0. It is comprised of three words. Word 0 is the status word, containing 15 status bits. Word 1 is the preset value. Word 2 is the accumulated value. Once assigned to the HSC instruction, C5:0 is not available as an address for any other counter instructions.



CU = Counter Up Enable Bit
CD = Counter Down Enable Bit
DN = High Preset Reached Bit
OV = Overflow Occurred Bit
UN = Underflow Occurred Bit
UA = Update High-Speed Counter Accumulator Bit
HP = Accumulator ≥ High Preset Bit①
LP = Accumulator ≤ Low Preset Bit
IV = Overflow Caused High-Speed Counter Interrupt Bit①
IN = Underflow Caused High-Speed Counter Interrupt Bit①
IH = High Preset Reached Caused Interrupt Bit①
IL = Low Preset Reached Caused Interrupt Bit①
PE = High-Speed Counter Interrupt Pending Bit①
LS = High-Speed Counter Interrupt Lost Bit①
IE = High-Speed Counter Interrupt Enable Bit①

① To access these bits, place your cursor on the instruction and press [F8], Data Monitor.

Counter preset and accumulated values are stored as signed integers.

Using Status Bits

The high-speed counter status bits are retentive. When the high-speed counter is first configured, bits 3–7, 14, and 15 are reset and bit 1 (IE) is set.

- **Counter Up Enable Bit CU** (bit 15) is used with all of the high-speed counter types. If the HSC instruction is true, the CU bit is set to one. If the HSC instruction is false, the CU bit is set to zero. Do not write to this bit.
- **Counter Down Enable Bit CD** (bit 14) is used with the Bidirectional Counters (modes 3–8). If the HSC instruction is true, the CD bit is set to one. If the HSC instruction is false, the CD bit is set to zero. Do not write to this bit.
- **High Preset Reached Bit DN** (bit 13) For the Up Counters (modes 1 and 2), this bit is an edge triggered latch bit. This bit is set when the high preset is reached. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction.

  The DN bit is a reserved bit for all other Counter options (modes 3–8).

- **Overflow Occurred Bit OV** (bit 12) For the Up Counters (modes 1 and 2), this bit is set by the controller when the high preset is reached if the DN bit is set.

  For the Bidirectional Counters (modes 3–8), the OV bit is set by the controller after the hardware accumulator transitions from 32,767 to –32,768. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction for both the up and bidirectional counters.

- **Underflow Occurred Bit UN** (bit 11) is a reserved bit for the Up Counters (modes 1 and 2). Do not write to this bit.

Note

For the Bidirectional Counters (modes 3–8), the UN bit is set by the controller when the hardware accumulator transitions from –32,768 to +32,767. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction.

- **Update High-Speed Counter Accumulator Bit UA** (bit 10) is used with an OTE instruction to update the instruction image accumulator value with the hardware accumulator value. (The HSC instruction also performs this operation each time the rung with the HSC instruction is evaluated as true.)
- **Accumulator ≥ High Preset Bit HP** (bit 9) is a reserved bit for all Up Counters (modes 1 and 2).

  For the Bidirectional Counters (modes 3–8), if the hardware accumulator becomes greater than or equal to the high preset, the HP bit is set. If the hardware accumulator becomes less than the high preset, the HP bit is reset by the controller. Do not write to this bit. (Exception – you can set or reset this bit during the initial configuration of the HSC instruction. See page 7–6 for more information.)

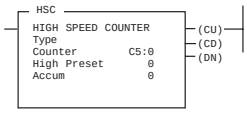
- **Accumulator ≤ Low Preset Bit LP** (bit 8) is a reserved bit for all Up Counters.

  For the Bidirectional Counters, if the hardware accumulator becomes less than or equal to the low preset, the LP bit is set by the controller. If the hardware accumulator becomes greater than the low preset, the LP bit is reset by the controller. Do not write to this bit. (Exception – you can set or reset this bit during the initial configuration of the HSC instruction. See page 7–6 for more information.)

- **Overflow Caused High-Speed Counter Interrupt Bit IV** (bit 7) is set to identify an overflow as the cause for the execution of the high-speed counter interrupt routine. The IN, IH, and IL bits are reset by the controller when the IV bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.
- **Underflow caused High-Speed Counter Interrupt Bit IN** (bit 6) is set to identify an underflow as the cause for the execution of the high-speed counter interrupt routine. The IV, IH, and IL bits are reset by the controller when the IN bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.
- **High Preset Reached Caused High-Speed Counter Interrupt Bit IH** (bit 5) is set to identify a high preset reached as the cause for the execution of the high-speed counter interrupt routine. The IV, IN, and IL bits are reset by the controller when the IH bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.
- **Low Preset Reached Caused High-Speed Counter Interrupt Bit IL** (bit 4) is set to identify a low preset reached as the cause for the execution of the high-speed counter interrupt routine. The IV, IN, and IH bits are reset by the controller when the IL bit is set. Examine this bit at the start of the high-speed counter interrupt routine (file 4) to determine why the interrupt occurred.
- **High-Speed Counter Interrupt Pending Bit PE** (bit 3) is set to indicate that a high-speed counter interrupt is waiting for execution. This bit is cleared by the controller when the high-speed counter interrupt routine begins executing. This bit is reset if an RAC or RES instruction is executed. Do not write to this bit.
- **High-Speed Counter Interrupt Lost Bit LS** (bit 2) is set if an high-speed counter interrupt occurs while the PE bit is set. You can reset this bit with an OTU instruction or by executing an RAC or RES instruction.
- **High-Speed Counter Interrupt Enable Bit IE** (bit 1) is set when the high-speed counter interrupt is enabled to run when an high-speed counter interrupt condition occurs. It is reset when the interrupt is disabled. This bit is also set when the high-speed counter is first configured. Do not write to this bit.

High-Speed Counter (HSC)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓					



Use this instruction to configure the high-speed counter. Only one HSC instruction can be used in a program. The high-speed counter is not operational until the first true execution of the HSC instruction. When the HSC rung is false, the high-speed counter is disabled from *counting*, but all other HSC features are operational.

The Counter address of the HSC instruction is fixed at C5:0. After the HSC is configured, the image accumulator (C5:0.ACC) is updated with the current hardware accumulator value every time the HSC instruction is evaluated as true or false.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Type** indicates the counter selected. Refer to page 7–7 for making your high-speed counter selection. Each type is available with reset and hold functionality.
- **High Preset** is the accumulated value that triggers a user-specified action such as updating outputs or generating an high-speed counter interrupt.
- **Accumulator** is the number of accumulated counts.

The following terminology is used in the following table to indicate the status of counting:

- Up↑ – increments by 1 when the input energizes (edge).
- Down↑ – decrements by 1 when the input energizes (edge).
- Reset↑ – resets the accumulator to zero when the input energizes (edge).
- Hold – disables the high-speed counter from counting while the input is energized (level).
- Count – increments or decrements by 1 when the input energizes (edge).
- Direction – allows up counts when the input is de-energized and down counts while the input is energized (level).
- A – input pulse in an incremental (quadrature) encoder (edge/level).
- B – input pulse in an incremental (quadrature) encoder (edge/level).
- Z – reset pulse in an incremental (quadrature) encoder (edge/level).
- ↑ – the signal is active on the rising edge only (off to on).

The table below lists the function key you press to choose the type of high-speed counter you want.

High-Speed Counter Type and Function Key	High-Speed Counter Functionality	Input Terminal Used			
		I/0	I/1	I/2	I/3
Up	Up Counter operation uses a single-ended input.	Up↑	Not Used	Not Used	Not Used
Up (with reset and hold)	Up Counter operation uses a single input with external reset and hold inputs.	Up↑	Not Used	Reset↑	Hold
Pulse and direction	Bidirectional operation uses both pulse and direction inputs.	Count↑	Direction	Not Used	Not Used
Pulse and direction (with external reset and hold)	Bidirectional operation uses both pulse and direction inputs with external reset and hold inputs.	Count↑	Direction	Reset↑	Hold
Up and down	Bidirectional operation uses both up and down direction inputs.	Up↑	Down↑	Not Used	Not Used
Up and down (with external reset and hold)	Bidirectional operation uses both up and down pulse inputs with external reset and hold inputs.	Up↑	Down↑	Reset↑	Hold
Encoder	Bidirectional operation uses quadrature encoder inputs.	A	B	Not Used	Not Used
Encoder (with external reset and hold)	Bidirectional operation uses both quadrature encoder inputs with external reset and hold inputs.	A	B	Z	Hold

One difference between Up Counters and Bidirectional Counters is that for Bidirectional Counters the accumulator and preset values are not changed by the high-speed counter when the presets are reached. The RAC and HSL instructions must be used for this function. The Up Counters clear the accumulator and re-load the high preset values whenever the preset is reached.

Using the Up Counter and the Up Counter with Reset and Hold

Up counters are used when the parameter being measured is uni-directional, such as material being fed into a machine or as a tachometer recording the number of pulses over a given time period.

Both types of Up Counters operate identically, except that the Up Counter with reset and hold uses external inputs 2 and 3.

For the Up Counter, each Off-to-On state change of input I:0/0 adds 1 to the accumulator until the high preset is reached. The accumulator is then automatically reset to zero. The Up Counter operates in the 0 to +32,767 range inclusive and can be reset to zero using the Reset (RES) instruction.

When the HSC instruction is first executed true, the:

- Accumulator C5:0.ACC is loaded to the hardware accumulator.
- High preset C5:0.PRE is loaded to the hardware high preset.

Operation

If you move data to the high preset without using the HSL instruction (with a MOV) after the high-speed counter has been configured, the data is loaded to the instruction image but is *not* loaded to the hardware. The modified high preset value is not loaded to the hardware until the existing hardware high preset is reached, or an RAC or RES instruction is executed.

The high preset value loaded to the hardware must be between 1 and 32,767 inclusive or an error `INVALID PRESETs LOADED TO HIGH SPEED COUNTER` (37H) occurs. Any value between –32,768 and +32,767 inclusive can be loaded to the hardware accumulator.

The Following Condition	Occurs when
A high preset is reached	either the hardware accumulator transitions from the hardware high preset –1 to the hardware high preset, or
	the hardware accumulator is loaded with a value greater than or equal to the hardware high preset, or
	the hardware high preset is loaded with a value that is less than or equal to the hardware accumulator.

When a high preset is reached, no counts are lost.

- Hardware and instruction accumulators are reset.
- Instruction high preset is loaded to the hardware high preset.
- If the DN bit is not set, the DN bit is set. The IH bit is also set and the IL, IV, and IN bits are reset.
- If the DN bit is already set, the OV bit is set. The IV bit is also set and the IL, IV and IN bits are reset.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IH bit is set and the IL, IV, and IN bits are set.

If the DN bit is already set when a high preset is reached, the OV bit is set.

The following tables summarize what the input state must be for the corresponding high-speed counter action to occur:

Up Counter

Input State					High-Speed Counter Action
Input Count (I/O)	Input Direction (I/1)	Input Reset (I/2)	Input Hold (I/3)	HSC Rung	
Turning Off-to-On	NA	NA	NA	True	Count Up
NA	NA	NA	NA	False	Hold Count

NA (Not Applicable)

Up Counter with Reset and Hold

Input state					High-Speed Counter Action
Input Count (I/O)	Input Direction (I/1)	Input Reset (I/2)	Input Hold (I/3)	HSC Rung	
Turning Off-to-On	NA	Off, On, or Turning Off	Off	True	Count Up
NA	NA	Off, On, or Turning Off	On	NA	Hold Count
NA	NA	Off, On, or Turning Off	NA	False	Hold Count
Off, On, or Turning Off	NA	Off, On, or Turning Off	NA	NA	Hold Count
NA	NA	Turning On	NA	NA	Reset to 0

NA (Not Applicable)

Using the Bidirectional Counter and the Bidirectional Counter with Reset and Hold

Bidirectional counters are used when the parameter being measured can either increment or decrement. For example, a package entering and leaving a storage bin is counted to regulate flow through the area.

The Bidirectional Counters operate identically except for the operation of inputs 1 and 0. For the Pulse and Direction type, input 0 provides the pulse and input 1 provides the direction. For the Up and Down type, input 0 provides the Up count and input 1 provides the Down count. Both types are available with and without reset and hold. Refer to page 7–7 for more information regarding Bidirectional Counter types.

For the Bidirectional Counters, both high and low presets are used. The low preset value must be less than the high preset value or an error `INVALID PRESETs LOADED TO HIGH SPEED COUNTER` (37H) occurs. Hardware low preset is set to –32,768.

Bidirectional Counters operate in the –32,768 to +32,767 range inclusive and can be reset to zero using the Reset (RES) instruction.

Operation

When the HSC instruction is first executed true, the:

- Instruction accumulator is loaded to the hardware accumulator.
- Instruction high preset is loaded to the hardware high preset.

After the first true HSC instruction execution, data can only be transferred to the hardware accumulator via an RES or RAC instruction, or to the hardware high and low presets via the HSL instruction.

Any instruction accumulator value between –32,768 and +32,767 inclusive can be loaded to the hardware.

The Following Condition	Occurs when
A high preset is reached	either the hardware accumulator transitions from the hardware high preset –1 to the hardware high preset, or
	the hardware accumulator is loaded with a value greater than or equal to the hardware high preset, or
	the hardware high preset is loaded with a value that is less than or equal to the hardware accumulator.

When a high preset is reached, the:

- HP bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IH bit is set and the IL, IV, and IN bits are reset.

Unlike the Up Counters, the accumulator value does not get reset and the high preset value does not get loaded from the image to the hardware high preset register.

The Following Condition	Occurs when
A low preset is reached	either the hardware accumulator transitions from the hardware low preset +1 to the hardware low preset, or
	the hardware accumulator is loaded with a value less than or equal to the hardware low preset, or
	the hardware low preset is loaded with a value that is greater than or equal to the hardware accumulator.

When the low preset is reached, the:

- LP bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IL bit is set and the IH, IV, and IN bits are reset.

An overflow occurs when the hardware accumulator transitions from +32,767 to –32,768. When an overflow occurs, the:

- OV bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IV bit is set and the IH, IL, and IN bits are reset.

An underflow occurs when the hardware accumulator transitions from –32,768 to +32,767. When an underflow occurs, the:

- UN bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IN bit is set and the IH, IL, and IV bits are reset.

The following tables summarize what the input state must be for the corresponding high-speed counter action to occur:

Bidirectional Counter (Pulse/direction)

Input State					High-Speed Counter Action
Input Count (I/0)	Input Direction (I/1)	Input Reset (I/2)	Input Hold (I/3)	HSC Rung	
Turning Off-to-On	Off	NA	NA	True	Count Up
Turning Off-to-On	On	NA	NA	True	Count Down
NA	NA	NA	NA	False	Hold Count

NA (Not Applicable)

Bidirectional Counter with Reset and Hold (Pulse/direction)

Input State					High-Speed Counter Action
Input Count (I/0)	Input Direction (I/1)	Input Reset (I/2)	Input Hold (I/3)	HSC Rung	
Turning Off-to-On	Off	Off, On, or Turning Off	Off	True	Count Up
Turning Off-to-On	On	Off, On, or Turning Off	Off	True	Count Down
NA	NA	Off, On, or Turning Off	NA	False	Hold Count
NA	NA	Off, On, or Turning Off	On	NA	Hold Count
Off, On, or Turning Off	NA	Off, On, or Turning Off	NA	NA	Hold Count
NA	NA	Turning On	NA	NA	Reset to 0

NA (Not Applicable)

Bidirectional Counter (Up/down count)

Input State			High-Speed Counter Action
Input Up Count (I/0)	Input Down Count (I/1)	HSC Rung	
Turning Off-to-On	Off, On, or Turning Off	True	Count Up
Off, On, or Turning Off	Turning Off-to-On	True	Count Down
NA	NA	False	Hold Count

NA (Not Applicable)

Bidirectional Counter with Reset and Hold (Up/down count)

Input State					High-Speed Counter Action
Input Up Count (I/0)	Input Down Count (I/1)	Input Reset (I/2)	Input Hold (I/3)	HSC Rung	
Turning Off-to-On	Off, On, or Turning Off	Off, On, or Turning Off	Off	True	Count Up
Off, On, or Turning Off	Turning Off-to-On	Off, On, or Turning Off	Off	True	Count Down
NA	NA	Off, On, or Turning Off	NA	False	Hold Count
NA	NA	Off, On, or Turning Off	On	NA	Hold Count
Off, On, or Turning Off	Off, On, or Turning Off	Off, On, or Turning Off	NA	NA	Hold Count
NA	NA	Turning On	NA	NA	Reset to 0

NA (Not Applicable)

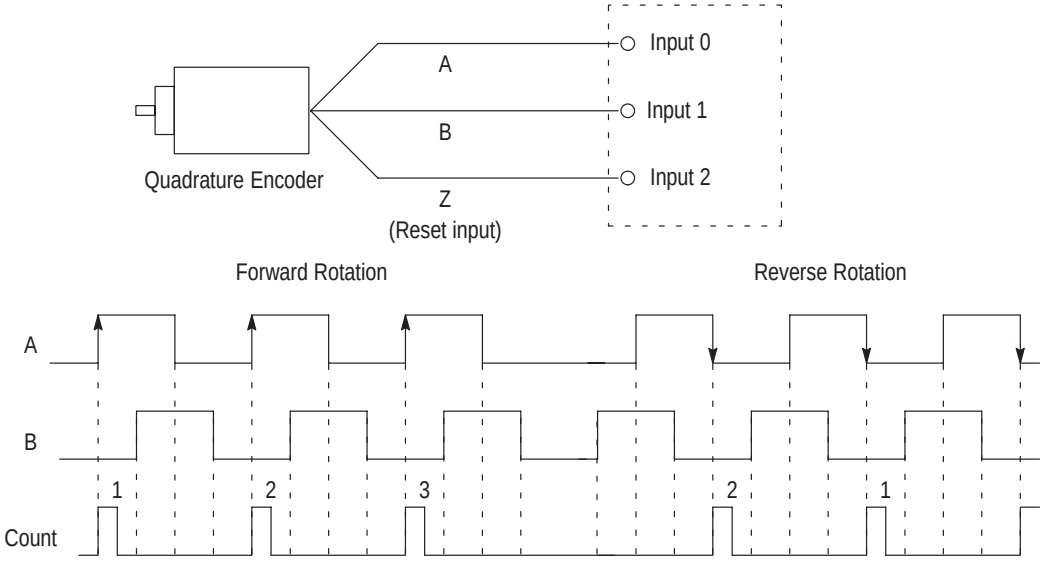
When up and down input pulses occur simultaneously, the high-speed counter counts up, then down.

Using the Bidirectional Counter with Reset and Hold with a Quadrature Encoder

The Quadrature Encoder is used for determining direction of rotation and position for rotating, such as a lathe. The Bidirectional Counter counts the rotation of the Quadrature Encoder.

Bidirectional Counters operate in the –32,768 to +32,767 range inclusive and can be reset to zero using the reset (RES) instruction. The following figure shows a quadrature encoder connected to inputs 0, 1, and 2. The count direction is determined by the phase angle between A and B. If A leads B, the counter increments. If B leads A, the counter decrements.

The counter can be reset using the Z input. The Z outputs from the encoders typically provide one pulse per revolution.



Operation

For the Bidirectional Counters, both high and low presets are used. The low preset value must be less than the high preset value or an error `INVALID PRESETs LOADED TO HIGH SPEED COUNTER` (37H) occurs.

When the HSC instruction is first executed true, the:

- Instruction accumulator is loaded to the hardware accumulator.
- Instruction high preset is loaded to the hardware high preset.

Any instruction accumulator value between –32,768 and +32,767 inclusive can be loaded to the hardware.

After the first true HSC instruction execution, data can only be transferred to the hardware accumulator via an RES or RAC instruction, or to the hardware high and low presets via the HSL instruction.

The Following Condition	Occurs when
A high preset is reached	either the hardware accumulator transitions from the hardware high preset –1 to the hardware high preset, or
	the hardware accumulator is loaded with a value greater than or equal to the hardware high preset, or
	the hardware high preset is loaded with a value that is less than or equal to the hardware accumulator.

When a high preset is reached, the:

- HP bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IH bit is set and the IL, IN, and IV bits are reset.

Unlike the Up Counters, the accumulator value does not reset and the high preset value does not get loaded from the image to the hardware high preset register.

The Following Condition	Occurs when
A low preset is reached	either the hardware accumulator transitions from the hardware low preset +1 to the hardware low preset, or
	the hardware accumulator is loaded with a value less than or equal to the hardware low preset, or
	the hardware low preset is loaded with a value that is greater than or equal to the hardware accumulator.

When a low preset is reached, the:

- LP bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IL bit is set and the IH, IN, and IV bits are reset.

An overflow occurs when the hardware accumulator transitions from +32,767 to –32,768. When an overflow occurs, the:

- OV bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IV bit is set and the IH, IL, and IN bits are reset.

An underflow occurs when the hardware accumulator transitions from –32,768 to +32,767. When an underflow occurs, the:

- UN bit is set.
- High-speed counter interrupt file (program file 4) is executed if the interrupt is enabled. The IN bit is set and the IH, IL, and IV bits are reset.

The following tables summarize what the input state must be for the corresponding high-speed counter action to occur:

Bidirectional Counter (Encoder)

Input State			High-Speed Counter Action
Input A (I/0)	Input B (I/1)	HSC Rung	
Turning On	Off	True	Count Up
Turning Off	Off	True	Count Down
NA	On	NA	Hold Count
NA	NA	False	Hold Count

NA (Not Applicable)

Bidirectional Counter with Reset and Hold (Encoder)

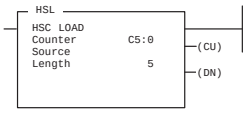
Input State					High-Speed Counter Action
Input A (I/0)	Input B (I/1)	Input Z (I/2)	Input Hold (I/3)	HSC Rung	
Turning On	Off	Off	Off	True	Count Up
Turning Off	Off	Off	Off	True	Count Down
Off or On	NA	Off	NA	NA	Hold Count
NA	On	Off	NA	NA	Hold Count
NA	NA	Off	NA	False	Hold Count
NA	NA	Off	On	NA	Hold Count
Off	Off	On①	NA	NA	Reset to 0

NA (Not Applicable)

① The optional hardware high-speed counter reset is the logical coincidence of $\overline{A} \times \overline{B} \times Z$.

High-Speed Counter Load (HSL)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓					



This instruction allows you to set the low and high presets, low and high output source, and the output mask. When either a high or low preset is reached, you can instantly update selected outputs.

If you are using the HSL instruction with the Up Counter, the high preset must be ≥ 1 and ≤ +32,767 or an error `INVALID PRESETs LOADED TO HIGH SPEED COUNTER` (37H) occurs. For the bidirectional counters, the high preset must be greater than the low preset or an error `INVALID PRESETs LOADED TO HIGH SPEED COUNTER` (37H) occurs.

The Counter referenced by this instruction has the same address as the HSC instruction counter and is fixed at C5:0.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is an address that identifies the first of five data words used by the HSL. The source can be either an integer or binary file element.
- **Length** is the number of elements starting from the source. This number is always 5.

Operation

The HSL instruction allows you to *configure* the high-speed counter to instantaneously and automatically update external outputs whenever a high or low preset is reached. The physical outputs are automatically updated in less than 30 µs. (The physical turn-on time of the outputs is *not* included in this amount.) The output image is then automatically updated at the next poll for user interrupts, IOM instruction, or output scan, whichever occurs first.

With this instruction, you can change the high preset for the up counters or both the high and low presets for Bidirectional Counters during run. You can also modify the output mask configuration during run.

The source address is either an integer or binary file element. For example, if N7:5 is selected as the source address, the additional parameters for the execution of this instruction would appear as shown in the following table.

Parameter Image Location	Up Counter Only	Bidirectional Counters	Description
N7:5	Output Mask	Output Mask	Identifies which group of bits in the output file (word 0) are controlled. 000F=bits 3–0 00F0=bits 7–4 0003=bits 0 and 1 00FF= bits 7–0
N7:6	Output Source	Output High Source	(Up count.) The status of bits in this word are written “through” the mask to the actual outputs.
N7:7	Hi Preset	High Preset	(Up count.) When the accumulator reaches this value, the output source is written through the output mask to the actual outputs, and the HSC subroutine (file 4) is scanned.
N7:8	Reserved	Output Low Source	(Down count.) The status of bits in this word are written “through” the mask to the actual outputs.
N7:9	Reserved	Low Preset	(Down count.) When the accumulator reaches this value, the output source is written through the output mask to the actual outputs, and the HSC subroutine (file 4) is scanned.

The bits in the output mask directly correspond to the physical outputs. If a bit is set to 1, the corresponding output can be changed by the high-speed counter. If a bit is set to 0, the corresponding output cannot be changed by the high-speed counter.

The bits in the high and low sources also directly correspond to the physical outputs. The high source is applied when the high preset is reached. The low source is applied when the low preset is reached. The final output states are determined by applying the output source over the mask and updating only the unmasked outputs (those with a 1 in the mask bit pattern).

You can always change the state of the outputs via the user program or programming device regardless of the output mask. The high-speed counter only modifies selected outputs and output image bits based on source and mask bit patterns when the presets are reached. The last device that changes the output image (i.e., user program or high-speed counter) determines the actual output pattern.

Forces override any output control from either the high-speed counter or from the output image. Forces may also be applied to the high-speed counter inputs. Forced inputs are recognized by the high-speed counter (e.g., a forced count input off and on increments the high-speed accumulator).

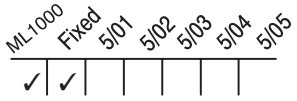
The high-speed counter hardware is updated immediately when the HSL instruction is executed regardless of high-speed counter type (Up Counter or Bidirectional Counter). For the Up Counters, the last two registers are ignored since the low preset does not apply.

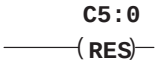
If a fault occurs due to the HSL instruction, the HSL parameters are not loaded to the high-speed counter hardware. You can use more than one HSL instruction in your program. The HSL instructions can have different image locations for the additional parameters.

Do not change a preset value *and* an output mask/source with the same HSL instruction as the accumulator is approaching the old preset value.

If the high-speed counter is enabled and the HSL instruction is evaluated true, the high-speed counter parameters in the HSL instruction are applied immediately without stopping the operation of the high-speed counter. If the same HSL instruction is being used to change the high-speed counter controlled mask/source and the preset, the mask/source is changed first and the preset second. (The preset is changed within 40μs after the mask/source.) If the original preset is reached after the new mask/source is applied but before the new preset is applied, the new outputs are applied immediately.

High-Speed Counter Reset (RES)





The RES instruction allows you to write a zero to the hardware accumulator and image accumulator.

The counter referenced by this instruction has the same address as the HSC instruction counter and is entered as C0.

Operation

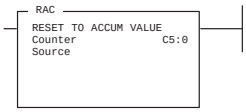
Execution of this instruction immediately:

- removes pending high-speed counter interrupts
- resets the hardware and instruction accumulators
- reset the PE, LS, OV, UN, and DN status bits
- loads the instruction high preset to the hardware high preset (if the high-speed counter is configured as an up counter)
- resets the IL, IH, IN, or IV status bits

You can have more than one RES instruction in your program.

High-Speed Counter Reset Accumulator (RAC)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓					



This instruction allows you to write a specific value to the hardware accumulator and image accumulator.

The counter referenced by this instruction has the same address as the HSC instruction counter and is fixed at C5:0.

Entering Parameters

Enter the following parameter when programming this instruction:

- **Source** represents the value that is loaded to the accumulator. The source can be a constant or an address.

Operation

Execution of the RAC:

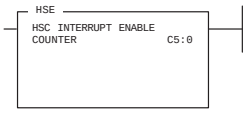
- removes pending high-speed counter interrupts
- resets the PE, LS, OV, UN, and DN status bits
- loads a new accumulator value to the hardware and instruction image
- loads the instruction high preset to the hardware high preset (if the high-speed counter is configured as an Up Counter)
- resets the IL, IH, IN, or IV status bits

The source can be a constant or any integer element in files 0–7. The hardware and instruction accumulators are updated with the new accumulator value immediately upon instruction execution.

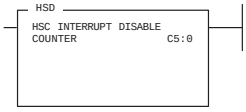
You can have more than one RAC instruction per program referencing the same source or different sources.

High-Speed Counter Interrupt Enable (HSE) and Disable (HSD)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓	✓					



These instructions enable or disable a high-speed counter interrupt when a high preset, low preset, overflow, or underflow is reached. Use the HSD and HSE in pairs to provide accurate execution for your application.



The counter referenced by these instructions have the same address as the HSC instruction counter and is fixed at C5:0.

Using HSE

Operation

When the high-speed counter interrupt is enabled, user subroutine (program file 4) is executed when:

- A high or low preset is reached.
- An overflow or underflow occurs.

When in Test Single Scan mode and in an idle condition, the high-speed counter interrupt is held off until the next scan trigger is received from the programming device. The high-speed counter accumulator counts while idle.

If the HSE is subsequently executed after the pending bit is set, the interrupt is executed immediately.

The default state of the high-speed counter interrupt is enabled (the IE bit set to 1).

If the high-speed counter interrupt routine is executing and another high-speed counter interrupt occurs, the second high-speed counter interrupt is saved but is considered pending. (The PE bit is set.) The second interrupt is executed immediately after the first one is finished executing. If an high-speed counter interrupt occurs while an high-speed counter interrupt is pending, the most recent high-speed counter interrupt is lost and the LS bit is set.

Using HSD

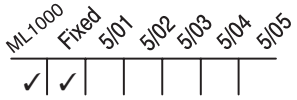
Operation

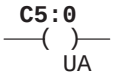
The HSD instruction disables the high-speed counter interrupt, preventing the interrupt subroutine from being executed.

This HSD instruction does not cancel an interrupt, but results in the pending bit (C5:0/3) being set when:

- A high or low preset is reached.
- An overflow or underflow occurs.

Update High-Speed Counter Image Accumulator (OTE)





An OTE bit instruction, when addressed for the high-speed counter (C5:0), causes the UA bit to be set. When this bit is set, the value in the hardware accumulator is written to the value in the image accumulator (C5:0.ACC). This provides you with real-time access to the hardware accumulator value. This is in addition to the automatic transfer from the hardware accumulator to the image accumulator that occurs each time the HSC instruction is evaluated.

Operation

This instruction transfers the hardware accumulator to the instruction accumulator. When the OTE/UA instruction is executed true, the hardware accumulator is loaded to the instruction image accumulator (C5:0.ACC).

What Happens to the HSC When Going to REM Run Mode

Once initialized, the HSC instruction retains its previous state when going through a mode change or power cycle. This means that the HSC Accumulator (C5:0.ACC) and High Preset values are retained. Outputs under the direct control of the HSC also retain their previous state. The Low Preset Reached and High Preset Reached bits (C0/LP and C0/HP) are also retained. They are examined by the HSC instruction during the high-speed counter's first true evaluation in the REM Run mode to differentiate a retentive REM Run mode entry from an external or initial Accumulator (C5:0.ACC) modification.

At the first true HSC instruction execution after going-to-run, the Low Preset is initialized to –32,768 and the output mask and high and low output patterns are initialized to zero. Use the HSL instruction during the first pass to restore any values necessary for your application.

You can modify the behavior of the high-speed counter at REM Run mode entry by adjusting the HSC parameters prior to the first true execution of the HSC instruction. The following example ladder rungs demonstrate different ways to adjust the HSC parameters.

Example 1

To enter the REM Run mode and have the HSC Outputs, ACC, and Interrupt Subroutine resume their previous state, apply the following:

```
(Rung 2:0)
No action required.  (Remember that all OUT instructions are
zeroed when entering the REM Run mode. Use SET/RST
instructions in place of OUT instructions in your conditional
logic requiring retention.)

|  S:1                                  +HSL---------------+ |
|--][-----------------------------------+HSC LOAD          +-|
|    15                                 |Counter       C5:0| |
|                                       |Source        N7:0| |
|                                       |Length           5| |
|                                       +------------------+ |

Rung 2:1
|                             +HSC---------------------+      |
|-----------------------------+HIGH SPEED COUNTER     +-(CU)-|
|                             |Type  Encoder(Res,Hld) +-(CD) |
|                             |Counter            C5:0+-(DN) |
|                             |High Preset        1000|      |
|                             |Accum                 0|      |
|                             +-----------------------+      |
```

Example 2

To enter the REM Run mode and retain the HSC ACC value while having the HSC Outputs and Interrupt Subroutine reassert themselves, apply the following:

```
Rung 2:0
Unlatch the C5:0/HP and C5:0/LP bits during the first scan BEFORE the
HSC instruction is executed for the first time.

|  S:1                                          +HSL---------------+  |
|--][-------------------------------------------+HSC LOAD          +- |
|    15                                         |Counter       C5:0|  |
|                                               |Source        N7:0|  |
|                                               |Length           5|  |
|                                               +------------------+ |

Rung 2:1
|  S:1                                                      C5:0     |
|--][-------------------------------------------------------+-(U)--+|--|
|    15                                                     |    HP |  |
|                                                           | C5:0  |  |
|                                                           +--(U)--+  |
|                                                               LP     |

Rung 2:2
|                                    +HSC--------------------+      |
|------------------------------------+HIGH SPEED COUNTER     +-(CU)-|
|                                    |Type  Encoder (Res,Hld)+-(CD) |
|                                    |Counter            C5:0+-(DN) |
|                                    |High Preset        1000|      |
|                                    |Accum                 0|      |
|                                    +-----------------------+      |
```

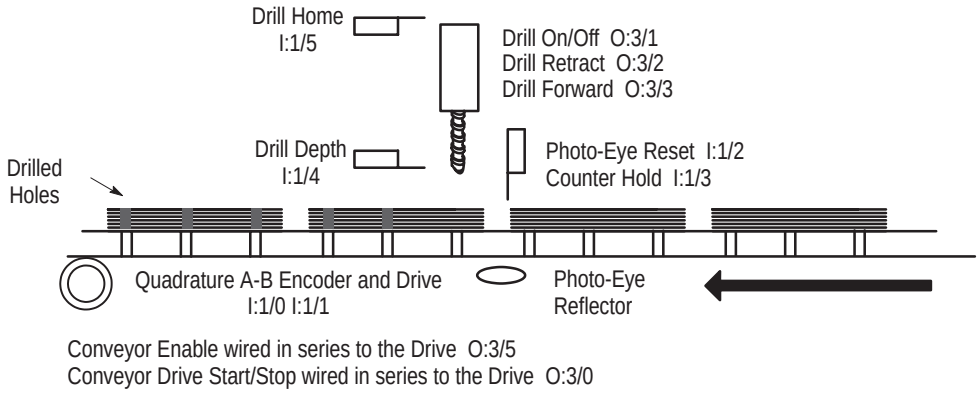
Example 3

To enter the REM Run mode and have the HSC ACC and Interrupt Subroutine resume their previous state, while externally initializing the HSC outputs, apply the following:

```
Rung 2:0
Unlatch or Latch the output bits under HSC control during the first
scan after the HSC instruction is executed for the first time. (Note:
you could place this rung before the HSC instruction; however, this is
not recommended.)

|  S:1                                          +HSL---------------+ |
|--][-------------------------------------------+HSC LOAD          +-|
|    15                                         |Counter       C5:0| |
|                                               |Source        N7:0| |
|                                               |Length           5| |
|                                               +------------------+ |

Rung 2:1
|                                    +HSC--------------------+      |
|------------------------------------+HIGH SPEED COUNTER     +-(CU)-|
|                                    |Type  Encoder (Res,Hld)+-(CD) |
|                                    |Counter            C5:0+-(DN) |
|                                    |High Preset        1000|      |
|                                    |Accum                 0|      |
|                                    +-----------------------+      |

Rung 2:2
This rung is programmed with the knowledge of an HSL mask of 0007
(outputs 0–2 are used) and initializes the HSC outputs each REM Run
mode entry. Outputs O/0 and O/1 are off, while Output O/2 is on.

|  S:1                                                      O:0      |
|--][-----------------------------------------------------+--(U)--+|--|
|    15                                                  |    0   |  |
|                                                        |  O:0   |  |
|                                                        +--(U)---+  |
|                                                        |    1   |  |
|                                                        |  O:0   |  |
|                                                        +--(L)---+  |
|                                                             2      |
```

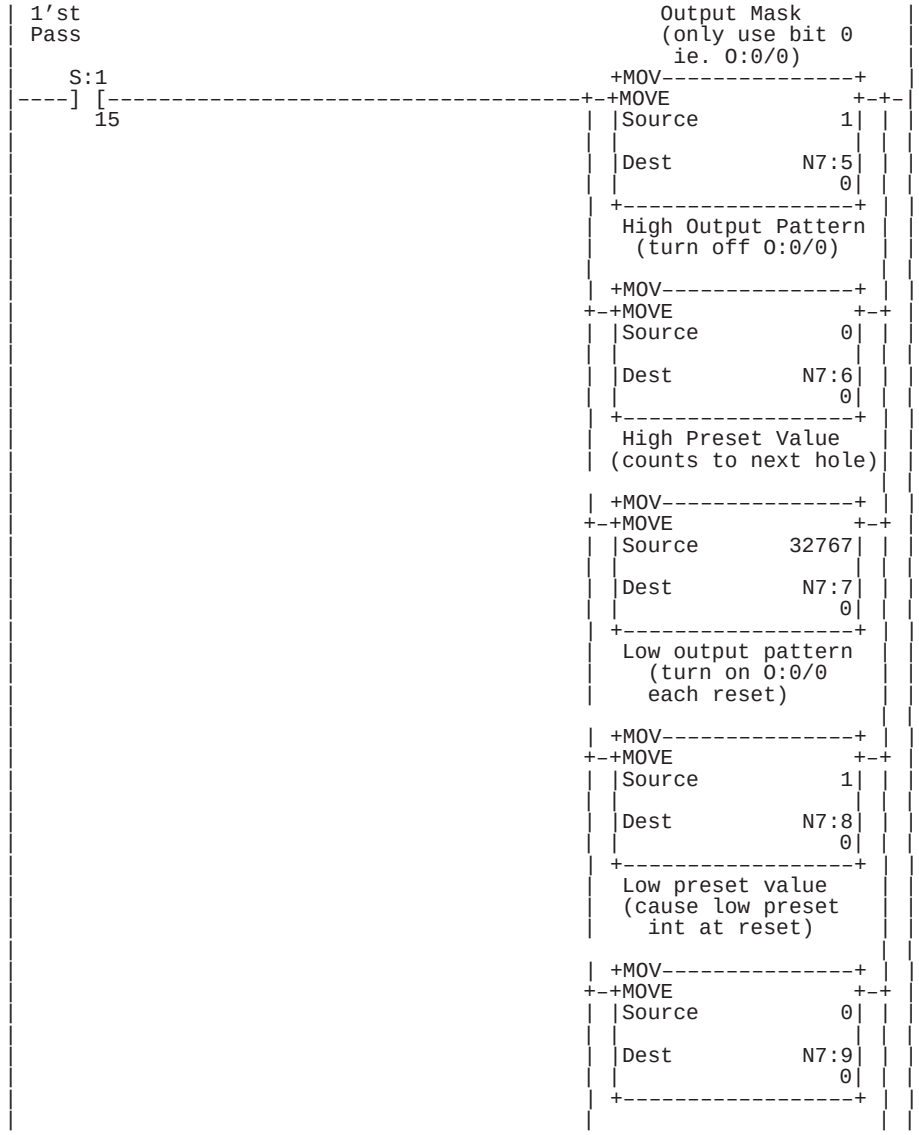
High-Speed Counter Instructions in the Paper Drilling Machine Application Example

The ladder rungs in this section demonstrate the use of the HSC instruction in the paper drilling machine application example started in chapter 4. Refer to appendix H for the complete paper drilling machine application example.



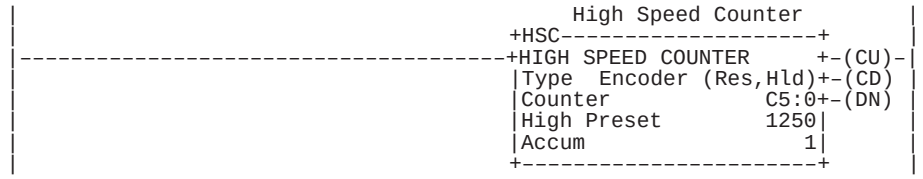
The main program file (file 2) initializes the HSC instruction, monitors the machine start and stop buttons, and calls other subroutines necessary to run the machine. Refer to the comments preceding each rung for additional information.

Rung 2:0
Initializes the high-speed counter each time the REM Run mode is entered. The high-speed counter data area (N7:5 – N7:9) corresponds with the starting address (source address) of the HSL instruction. The HSC instruction is disabled each entry into the REM run mode until the first time that it is executed as true. (The high preset was ”pegged” on initialization to prevent a high preset interrupt from occurring during the initialization process.)

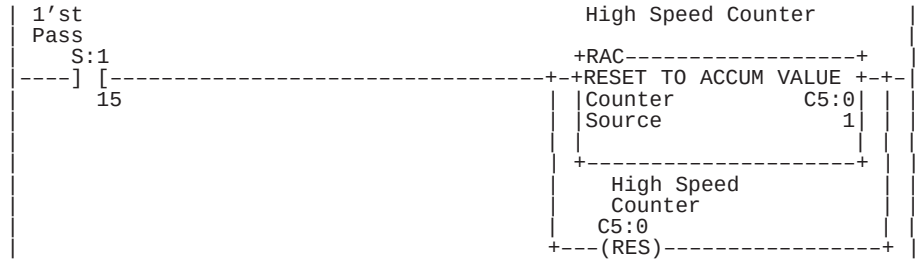


Rungs 2.0 and 2.2 are required to write several parameters to the high-speed counter data file area. These two rungs are conditioned by the first pass bit during one scan when the controller is going from REM program to REM Run mode.

Rung 2:1
This HSC instruction is not placed in the high-speed counter interrupt subroutine. If this instruction were placed in the interrupt subroutine, the high-speed counter could never be started or initialized (because an interrupt must first occur in order to scan the high-speed counter interrupt subroutine).



Rung 2:2
Forces a high-speed counter low preset interrupt to occur each REM Run mode entry. An interrupt can only occur on the transition of the high-speed counter accum to a preset value (accum reset to 1, then 0). This is done to allow the high-speed counter interrupt subroutine sequencers to initialize. The order of high-speed counter initialization is: (1)load high-speed counter parameters (2)execute HSL instruction (3)execute true HSC instruction (4)(optional) force high-speed counter interrupt to occur.



The high-speed counter is used to control the conveyer position. The high-speed counter counts pulses supplied by the conveyer's encoder via hardware inputs I:0/0 and I:0/1. Hardware inputs I:0/2 (reset) and I:0/3 (hold) are connected to a photo-switch ensuring the HSC instruction only counts encoder pulses when a manual is in front of the drill and that the high-speed counter is reset at the leading edge of each manual.

The high-speed counter clears the conveyer drive output bit (O:0/0) each time a high preset is reached. As a result, the drive decelerates and stops the conveyer motor. The high-speed counter clears the output within microseconds ensuring accuracy and repeatability.

The high-speed counter sets the conveyer drive output bit (O:0/0) each time a low preset is reached. As a result, the drive accelerates and maintains the conveyer motor.

When the manual has travelled the specified distance set by the high-speed counter high preset value, the high-speed counter interrupt subroutine signals the main program to perform the drilling sequence. For more information regarding the interrupt subroutine used in this program, refer to the application example in chapter 12.

This example uses the Quadrature Encoder with reset and hold instruction. The high-speed counter accumulator increments and decrements based on the quadrature relationship of the encoder's A and B inputs (I:0/0 and I:0/1). The accumulator is cleared to zero when the reset is activated or when the RES instruction is executed. All presets are entered as a relative offset to the leading edge of a manual. The presets for the hole patterns are stored in the SQO instructions. (Refer to chapter 6 for the SQO instruction.) The high-speed counter external reset input (I:0/2) and the external hold input (I:0/3) are wired in parallel to prevent the high-speed counter from counting while the reset is active.

The input filter delays for both the high-speed counter A and B inputs (I:0/0 and I:0/1) as well as the high-speed counter reset and hold inputs (I:0/2 and I:0/3) can be adjusted.

```
Rung 4:5
Interrupt occurred due to low preset reached.

|  C5:0                                          +RET---------------+-|
|----][------------------------------------------+RETURN            + |
|      IL                                        +------------------+ |
Rung 4:6
Signals the main program (file 2) to initiate a drilling sequence.  The
high-speed counter has already stopped the conveyor at the correct
position using its high preset output pattern data (clear O:0/0).  This
occurred within microseconds of the high preset being reached (just
prior to entering this high-speed counter interrupt subroutine).  The
drill sequence subroutine resets the drill sequence start bit and sets
the conveyor drive bit (O:0/0) upon completion of the drilling
sequence.

| interrupt occurred         |                  Drill Sequence Start |
| due to high preset reached |                                       |
|   C5:0                                                    B3      |
|----] [-------------------------------------------------------(L)-----|
|      IH                                                     32    |
Rung 4:7

|                                                                    |
|-------------------------------+END+-------------------------------|
|                                                                    |
```

8 SLC Communication Instructions

This chapter contains general information about the SLC communication instructions. Each of the instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction
- an application example and timing diagrams

Communication Instructions

Instruction		Purpose	Page
Mnemonic	Name		
MSG	Message Read/Write	This instruction transfers data from one node to another on the communication network. When the instruction is enabled, message transfer is pending. Actual data transfer takes place at the end of scan.	**8–2**
SVC	Service Communications	When conditions preceding the SVC instruction in the rung are true, the SVC instruction interrupts the program scan to execute the service communication portion of the operating cycle.	**8–58**

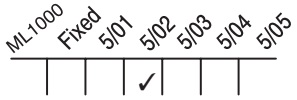
About the Communication Instructions

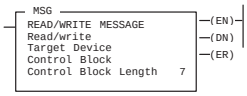
Use the Message instruction to send and receive data from other processors and devices. Use the SVC instruction to enhance communication performance of your processor.

In this chapter you will find a general overview preceding each type of instruction:

- Message instruction for the SLC 5/02 processor
- Message instruction for the SLC 5/03 and SLC 5/04 processors
- Message instruction for the SLC 5/05 processor
- Service Communication instruction for the SLC 5/02 processor
- Service Communication instruction for SLC 5/03 and higher processors

SLC 5/02 Message Instruction Overview





Output Instruction

This is an output instruction that allows you to transfer data from one node to another via the DH-485 network. The SLC 5/02 processor can service one message instruction at any given time, although the processor may hold several messages "enabled and waiting." Waiting messages are serviced one at a time in sequential order (first in first out).

Operation

The instruction can be programmed as a read or write message. The target device can be another SLC 500 processor on the network, or a non-SLC 500 device, using the common interface file (485CIF file 9 in SLC 500 processors). The 485CIF protocol is also used for PLC-2 type messages.

The data associated with a message write instruction is *not* sent when you enable the instruction. Rather, it is sent at the end of the scan or at the time a Service Communication (SVC) or Refresh (REF) instruction in your ladder program is enabled. In some instances, this means that you must buffer data in your application.

When you select the SLC500 as your target device, communication can take place between the SLC 5/02 processor and any other SLC 500 family processor.

Related Status File Bits

Three status file bits are related to the MSG instruction:

- **Incoming Command Pending Bit (S:2/5)** – This bit is set when the processor determines that another node on the network has requested information or supplied a command to it. This bit can become set at any time. This bit is cleared when the processor services the request (or command).

  Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.

- **Message Reply Pending Bit (S:2/6)** – This bit is set when another node on the network has supplied the information that you requested in the MSG instruction of your processor. This bit is cleared when the processor stores the information and updates your MSG instruction.

  Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.

- **Outgoing Message Command Pending Bit (S:2/7)** – This bit is set when one or more messages in your program are enabled and waiting, but no message is being transmitted at the time. As soon as transmission of a message begins, the bit is cleared. After transmission, the bit is set again if there are further messages waiting, or it remains cleared if there are no further messages waiting.

  Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.

You may also want to use bit S:2/15 Communications Servicing Selection. Refer to appendix B in this manual for more information.

Available Configuration Options

The following configuration options are available with a SLC 5/02 processor:

- Peer-to-Peer Read/Write on a Local network to another SLC 500 processor
- Peer-to-Peer Read/Write on a Local network to a 485CIF (PLC2 emulation)

Refer to appendix E for valid parameters when programming the Message instruction.

Entering Parameters

After you place the MSG instruction on a rung, specify whether the message is to be a read or write. Then specify the target device and the control block for the MSG instruction.

- **Read/Write** – Read indicates that the local processor (processor in which the instruction is located) is receiving data; write indicates that the processor is sending data.
- **Target Device** identifies the type of device which will receive data. Valid options are:
  - 500CPU, if the target device is another SLC processor
  - 485CIF, if the target device is a non-SLC device (PLC2 emulator)
- **Control Block** is an integer file address that you select. It is a 7-element file, containing the status bits, target file address, and other data associated with the message instruction.
- **Control Block Length** is fixed at seven elements. This field cannot be altered.

Note

The MSG control block length increases from 7 to 14 words when changing from a SLC 5/02 to a SLC 5/03, SLC 5/04 (channel 0, DH-485), or SLC 5/05 (channel 0, DH-485) processor program. Make sure that there are at least 7 unused words following each MSG control block in your program.

Using Status Bits

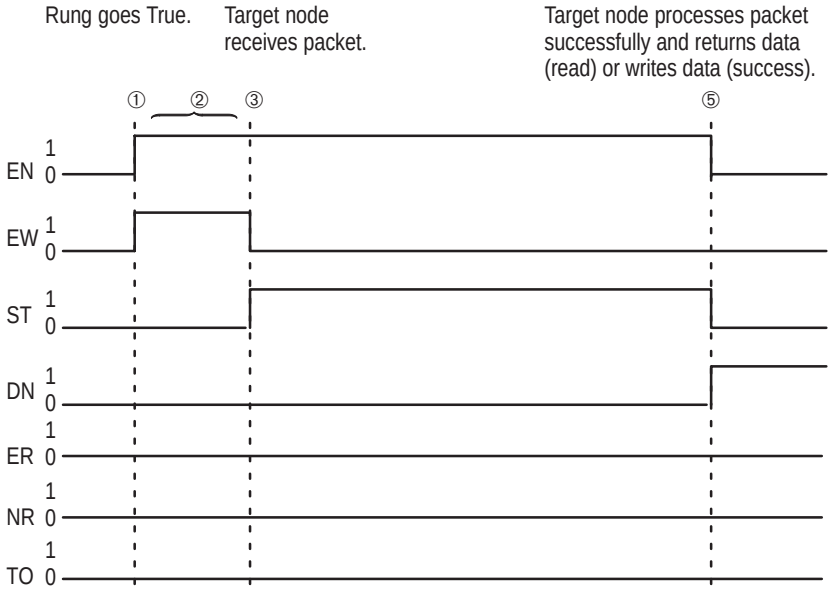
```
Read/Write:                          READ          ignore if timed out:   0  TO
 Target Device:                     500CPU                to be retried:   0  NR
 Control Block:                       N7:0           awaiting execution:   0  EW
 Local Destination File Address:       ***
 Target Node:                            0                        error:   0  ER
 Target File Address:                  ***                message done:   0  DN
 Message Length in elements            ***         message transmitting:   0  ST
                                                        message enabled:   0  EN

                                                    control bit address:    N7:0/8

 ERROR CODE:  0
 Error Code Desc:
```

The right column in the display above lists the various status bits associated with the SLC 5/02 MSG instruction.

- **Time Out Bit TO (bit 08)** You can set this bit in your application to remove an active message instruction from processor control. Your application must supply its own timeout value. An example appears on page 8–13.
- **No Response Bit NR (bit 09)** is set if the target processor does not respond to the first message request. The NR bit is reset when the ER, DN, or ST bit is set.
- **Enabled and Waiting Bit EW (bit 10)** is set after the enable bit is set and the message is waiting to be sent.
- **Error Bit ER (bit 12)** is set when message transmission has failed. The ER bit is reset the next time the associated rung goes from false to true.
- **Done Bit DN (bit 13)** is set when the message is transmitted successfully. The DN bit is reset the next time the associated rung goes from false to true.
- **Start Bit ST (bit 14)** is set when the processor receives acknowledgement from the target device. The ST bit is reset when the DN, ER, or TO bit is set.
- **Enable Bit EN (bit 15)** is set when rung conditions go true and the instruction is being executed. It remains set until the message transmission is complete and the rung goes false.

Timing Diagram for SLC 5/02 MSG Instruction

The following section describes the timing diagram for a SLC 5/02 MSG instruction.



1. When the MSG rung becomes true and the MSG is scanned, the **EN** bit is set and remains set until either the **DN**, **ER**, or **TO** bit is set. The **EW** bit is set, indicating that the MSG instruction has been placed in the MSG Queue. (The SLC 5/02 processor always has room in the MSG Queue.) The queue works on a first-in-first-out basis that allows the SLC 5/02 processor to remember the order the MSG instructions were enabled. Note that the program does not have access to the SLC 5/02 MSG Queue.

2. At the next end of scan or Service Communication instruction (SVC), the SLC 5/02 processor determines if it should examine the MSG Queue for "something to do." The processor bases its decision on the state of bit S:2/15, DH-485 communication requests from other nodes, and if a previous MSG instruction is already in progress. If the 5/02 processor determines that it should not access the queue, the **EN** and **EW** bits remain set until the next end of scan or SVC.

If the SLC 5/02 processor determines that it has “something to do” it uses the first message queue entry to build a DH-485 packet. If a packet can be successfully built, it is placed in the transmit buffer. If a packet cannot be successfully built, the **ER** bit is set and a code is placed in the MSG block to inform you of the error.

If this were a MSG Write instruction, the source data would be transferred to the transmit buffer at this time.

The SLC 5/02 processor then exits the end of scan or SVC portion of the scan. The processor’s background communication function sends the transmitted buffer packet to the Target Node that you specified in your MSG instruction.

3. If the Target Node successfully receives the DH-485 packet, it sends back an ACK (an acknowledge). The ACK causes the processor to clear the **EW** bit and set the **ST** bit. Note that the Target Node has not yet examined the DH-485 packet to see if it understands your request.

Once the **ST** bit is set, the processor waits indefinitely for a reply from the Target Node. The Target Node is not required to respond within any give time frame. At this time, no other MSG instruction will be serviced.

Note

*If the Target Node faults or power cycles during this time frame of a MSG transaction, you will never receive a reply. This is why it is recommended you use a timer instruction in conjunction with the **TO** bit. Refer to the example on page 8–13.*

Step 4 is not shown in the timing diagram.

4. If you do not receive an ACK, step 3 does not occur. Instead a NAK (no acknowledge) is received. When this happens, the **ST** bit remains clear. A NAK indicates:

- either the Target Node is not there,
- it does not respond
- it is too busy, or
- it receives a corrupt DH-485 packet.

When a NAK occurs, the **EW** bit is cleared and the **NR** bit is set for one scan. The next time the MSG instruction is scanned, the **ER** bit is set and the **NR** bit is cleared. This indicates that the MSG instruction failed. Note that if the Target Node is too busy, the **ER** bit is not set. Instead, the MSG instruction re-queues itself for re-transmission.

5. Following the successful receipt of the packet, the Target Node sends a reply packet. The reply packet will contain one of the following responses:

   - I have successfully performed your write request.
   - I have successfully performed your read request, and here is your data.
   - I have not performed your request, you are in error.

   At the next end of scan or SVC, following the Target Node's reply, the SLC 5/02 processor examines the DH-485 packet from the target device. If the reply contains "I have successfully performed your write request," the **DN** bit is set and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

   If the reply contains "I have successfully performed your read request, and here is your data," the data is written to the data table, the **DN** bit is set and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

   If the reply contains "I have not performed your request, you are in error," the **ER** bit is set and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

Control Block Layout

The control block layout is shown below if you select a 500CPU as the target device:

Control Block Layout – 500CPU

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	Word
EN	ST	DN	ER		EW	NR	TO	Error Code								0
Node Number																1
Reserved for length in words																2
File Number																3
File Type (S, B, T, C, R, N)																4
Element Number																5
Reserved																6

The control block layout is shown below if you select a 485 CIF as the target device:

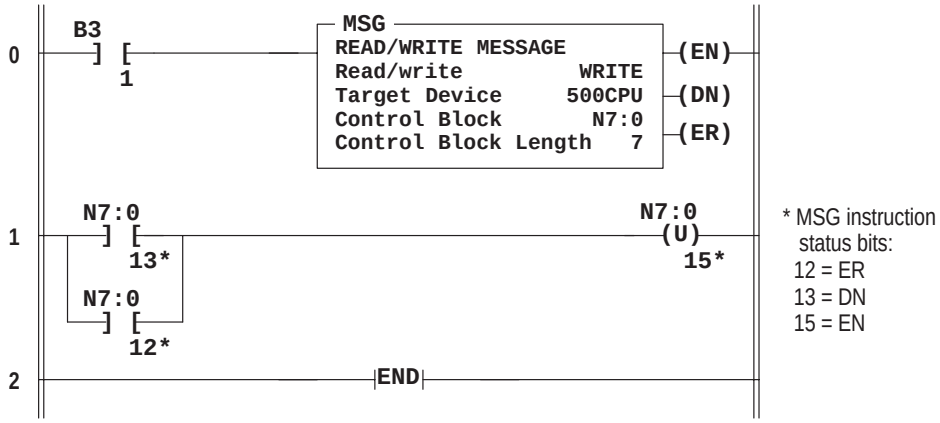
Control Block Layout – 485 CIF

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	Word
EN ST DN ER EW NR TO | Error Code	0
| Node Number	1
Offset elements	2
Reserved for length in elements	3
Not used	4
Not used	5
Not used	6

Application Examples for SLC 5/02 Processors

Example 1

Application example 1 shows how you can implement continuous operation of a message instruction.



Operation Notes

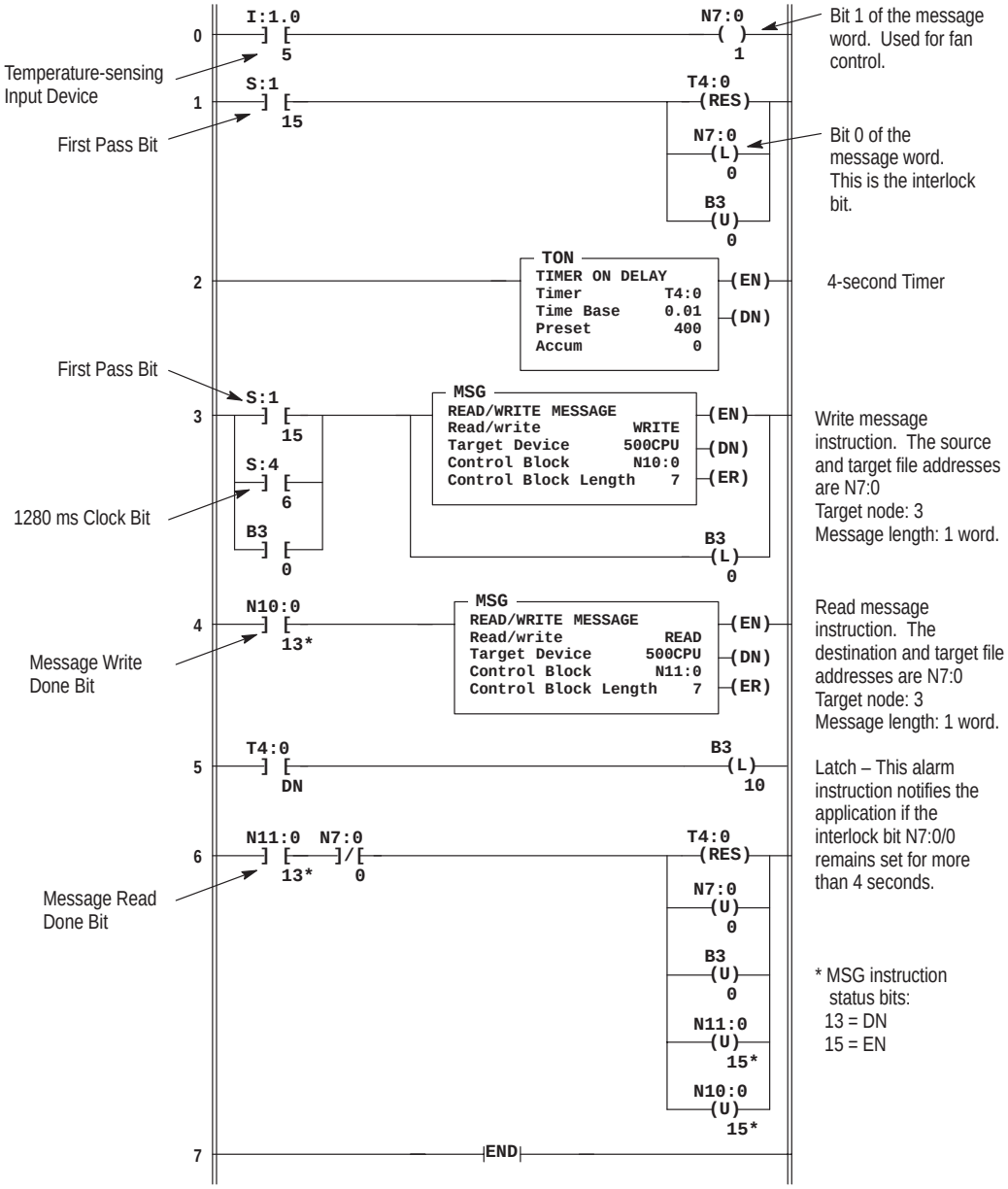
Bit B3/1 enables the MSG instruction. When the MSG instruction done bit is set, it unlatches the MSG enable bit so that the MSG instruction is enabled in the next scan. This provides continuous operation.

The MSG error bit also unlatches the enable bit. This provides continuous operation regardless of errors.

Example 2 – Program File 2 of SLC 5/02 Processor

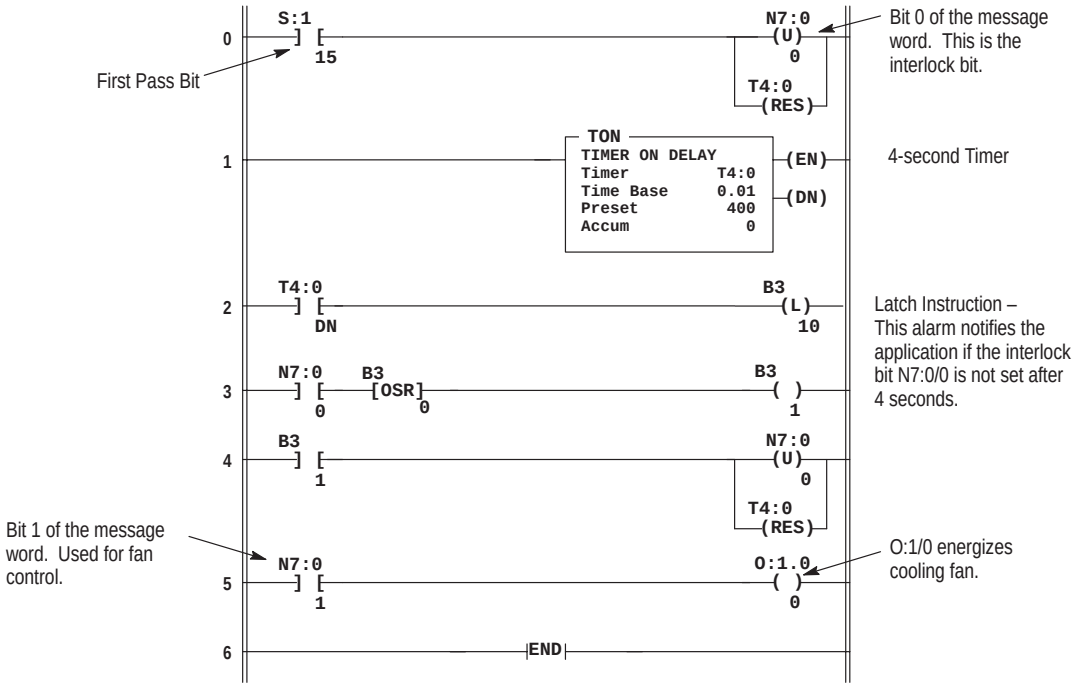
This example involves an SLC 5/02 processor and an SLC 5/01 processor communicating on a DH-485 network. Interlocking is provided to verify data transfer and to shut down both processors if communication fails.

A temperature-sensing device, connected as an input to the SLC 5/02 processor, controls the on-off operation of a cooling fan, connected as an output to the SLC 5/01 processor. The SLC 5/02 and SLC 5/01 ladder programs are explained in the figure on page 8–12.



Operation notes appear on the following page.

Program File 2 of SLC 5/01 Processor at Node 3



Operation Notes, SLC 5/02 and SLC 5/01 programs

Message instruction parameters: N7:0 is the message word. It is the target file address (SLC 5/01 processor) and the local source and destination addresses (SLC 5/02 processor) in the message instructions.

N7:0/0 of the message word is the interlock bit; it is written to the 5/01 processor as a 1 (set) and read from the SLC 5/01 processor as a 0 (reset).

N7:0/1 of the message word controls cooling fan operation; it is written to the SLC 5/01 processor as a 1 (set) if cooling is required or as a 0 (reset) if cooling is not required. It is read from the SLC 5/01 processor as either 1 or 0.

Word N7:0 should have a value of 1 or 3 during the message write execution. N7:0 should have a value of 0 or 2 during the message read execution.

Program initialization: The first pass bit S:1/15 initializes the ladder programs on run mode entry.

SLC 5/02 processor: N7:0/0 is latched; timer T4:0 is reset; B3/0 is unlatched (rung 1), then latched (rung 3). SLC 5/01 processor: N7:0/0 is unlatched; timer T4:0 is reset.

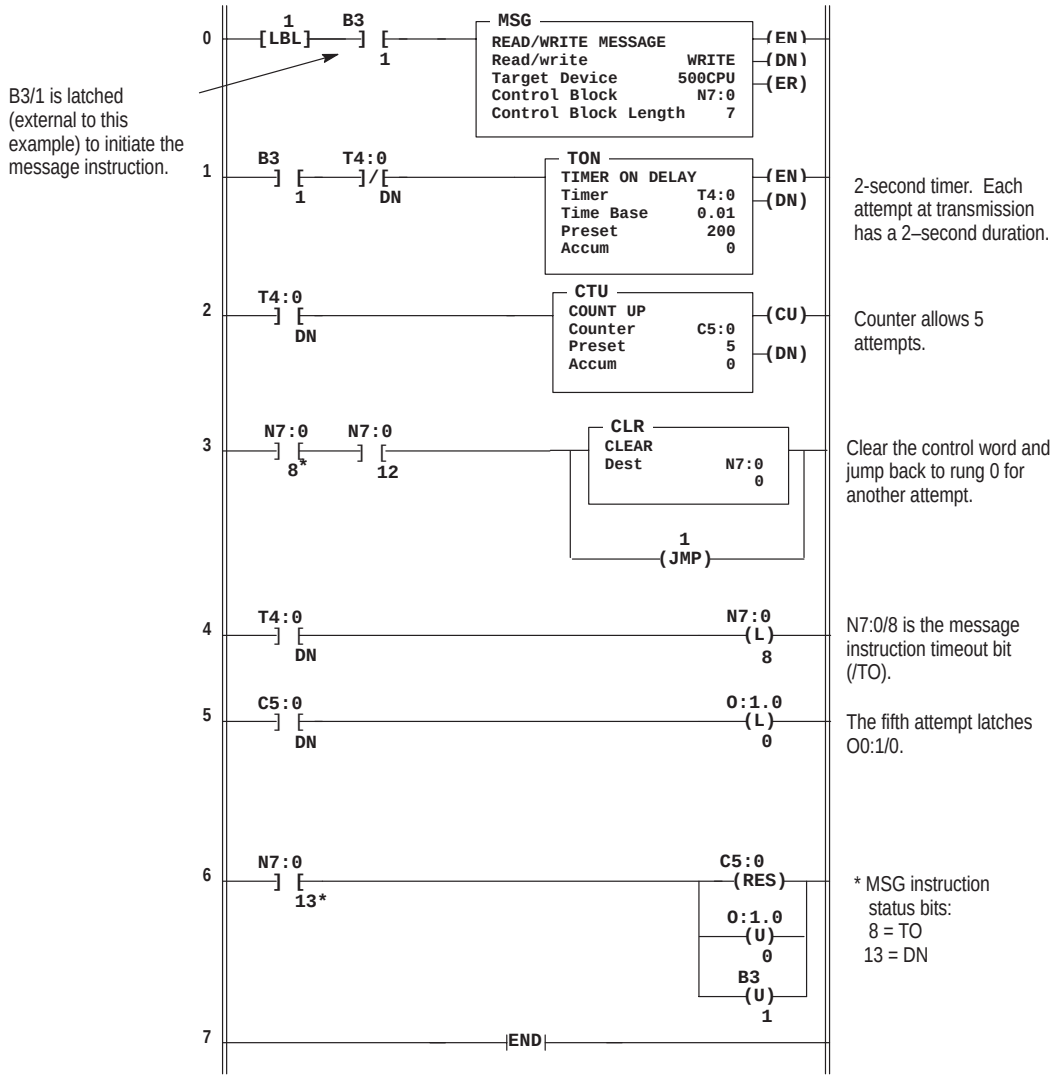
Message instruction operation: The message write instruction in the SLC 5/02 processor is initiated every 1280 ms by clock bit S:4/6. The done bit of the message write instruction initiates the message read instruction.

B3/0 latches the message write instruction. B3/0 is unlatched when the message read instruction done bit is set, provided that the interlock bit N7:0/0 is reset.

Communication failure: In the SLC 5/02 processor, bit B3/10 becomes set if interlock bit N7:0/0 remains set (1) for more than 4 seconds. In the SLC 5/01 processor, bit B3/10 becomes set if interlock bit N7:0/0 remains set (1) for more than 4 seconds. Your application can detect this event, take appropriate action, then unlatch bit B3/10.

Example 3

This example shows you how to use the timeout bit to disable an active message instruction. In this example, an output is energized after five unsuccessful attempts (two seconds duration) to transmit a message.



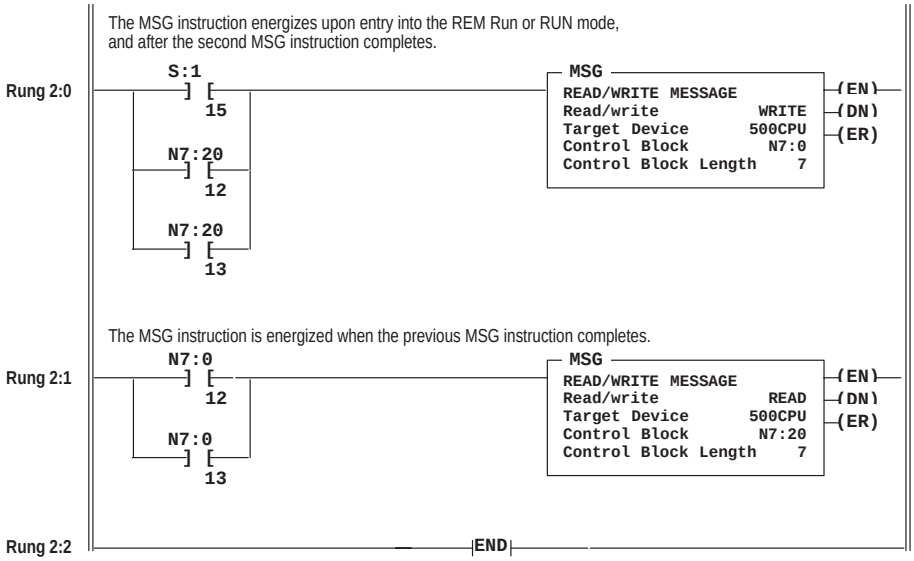
Operation Notes

The timeout bit is latched (rung 4) after a period of 2 seconds. This clears the message instruction from processor control on the next scan. The message instruction is then re-enabled for a second attempt at transmission. After 5 attempts, O:1/0 is latched.

A successful attempt at transmission resets the counter, unlatches O:1/0, and unlatches B3/1.

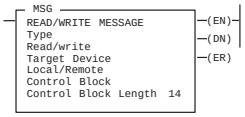
Example 4

This example shows you how to link message instructions together to transmit serially, one after another. In this example a MSG Write is followed by a MSG Read which causes the serial transmission.



SLC 5/03 and SLC 5/04 Message Instruction Overview

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	



Output Instruction

Data associated with a message write instruction is buffered when you enable the instruction. The SLC 5/03 with OS300 has four transmit buffers. The SLC 5/03 (OS301 and higher) and the SLC 5/04 processors service up to four message instructions per channel, for a maximum of eight message instructions.

To invoke the MSG instruction, toggle the MSG instruction rung from false-to-true. Do not toggle the rung again until the MSG instruction has successfully or unsuccessfully completed the previous message, indicated by the processor setting either the DN or ER bit.

Operation

SLC 5/03 OS300 – If a MSG instruction has entered one of the four "channel independent" transmission buffers and is waiting to be transmitted, its control block will have status bits EN and EW set. If more than four MSG instructions are enabled at one time, a "channel dependent" overflow queue is used to store the MSG instruction header blocks (not the data for a MSG write) from the fifth instruction to the fourteenth.

SLC 5/03(OS301 and higher) and SLC 5/04 – If a MSG instruction has entered one of the four "channel dependent" transmission buffers and is waiting to be transmitted, its control block will have status bits EN and EW set. If more than four MSG instructions for that channel are enabled at one time, a "channel dependent" overflow queue is used to store the MSG instruction header blocks (not the data for a MSG write) from the fifth instruction to the fourteenth. These instructions, queued in a FIFO order, will only have control block status bit EN set.

If more than 14 MSG instructions are enabled at one time for any one channel, only control block status bit WQ is set, as there is no room available to currently queue the instruction. This instruction must be re-scanned with true rung conditions until space exists in the overflow queue.

Note *If you consistently enable more MSG instructions than the buffers and queues can accomodate, the order in which MSG instructions enter the queue is determined by the order in which they are scanned. This means MSG instructions closest to the beginning of the program enter the queue regularly and MSG instructions later in the program may not enter the queue.*

You can use the timeout control similar to the SLC 5/02 MSG instruction or use the built in timeout control (recommended). If the timeout value is set to 0, which is the default, the functionality is similar to the SLC 5/02 MSG instruction. It differs in that once the TO bit is set, it will be reset automatically along with the ER bit on the next MSG rung false-to-true transition. We highly recommend setting the internal timeout value to something other than zero.

When using an SLC 5/03 or SLC 5/04 processor the message instruction:

- initiates reads and writes through RS-232 Channel 0 when configured for the following protocols:
  - DF1 Full-Duplex (Point-to-Point)
  - DF1 Half-Duplex Master/Slave (Point-to-Multipoint)
  - DH-485
- initiates reads and writes through:
  - DH-485 channel 1 (SLC 5/03 processors only)
  - DH+ channel 1 (SLC 5/04 processors only)

Related Status File Bits

Channel 1		Channel 0	
S:2/5	Incoming Command Pending Bit	S:33/0	Incoming Command Pending
S:2/6	Message Reply Pending Bit	S:33/1	Message Reply Pending
S:2/7	Outgoing Message Command Pending Bit	S:33/2	Outgoing Message Command Pending
S:2/15	Communications Servicing Selection Bit	S:33/5	Communications Servicing Selection
S:33/7	Message Servicing Selection	S:33/6	Message Servicing Selection

Refer to appendix B in this manual for more information on the above status file bits.

Available Configuration Options

The following configuation options are available when using an SLC 5/03 or SLC 5/04 processor. Refer to appendix E for valid parameters when programming the Message instruction.

- Peer-to-Peer Read/Write on a Local network to another SLC 500 processor
- Peer-to-Peer Read/Write on a Local network to a 485CIF
- Peer-to-Peer Read/Write on a Local network to a PLC-5 ①
- Peer-to-Peer Read/Write on a Remote network to another SLC 500 processor
- Peer-to-Peer Read/Write on a Remote network to a 485CIF (PLC2 emulation)
- Peer-to-Peer Read/Write on a Remote network to a PLC-5 processor ①

① Applies to SLC 5/03 (OS301 and higher) and SLC 5/04 processors.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Read/Write** – Read indicates that the local processor (processor in which the instruction is located) is receiving data; write indicates that it is sending data.
- **Target Device** identifies the type of device which will receive data. Valid options are:
  - 500CPU, if the target device is another SLC processor
  - 485CIF, if the target device is a non–SLC device on the DH-485 network
  - PLC-5, if the target device accepts PLC-5 commands
- **Local or Remote** identifies if the message is sent to a device on a local DH-485 or DH+ network, or to a remote device on another network through a bridge. Valid options are:
  - Local, if the target device is on the local network
  - Remote, if the target device is on a remote network
- **Control Block** is an integer file address that you select. It is a 14-word integer file, containing the status bits, target file address, and other data associated with the message instruction.
- **Control Block Length** is fixed at 14 elements. This field cannot be altered.

Note

The MSG control block length increases from 7 to 14 words when changing from a SLC 5/02 to a SLC 5/03 or SLC 5/04 (channel 0, DH-485) processor program. Make sure that there are at least 7 unused words following each MSG control block in your program.

Using Status Bits

The right column in the display below lists the various status bits associated with the SLC 5/03 and SLC 5/04 MSG instruction.

```
Type:                              Peer-to-Peer
Read/Write:                              READ                ignore if timed out:  0 TO
Target Device:                          500CPU                      to be retried:  0 NR
Local/Remote:                            Local               awaiting execution:  0 EW
Control Block:                           N10:0                    continuous run:  0 CO
Channel:                                     1                             error:  0 ER
Target Node:                                 2                      message done:  0 DN
                                                          message transmitting:  0 ST
                                                               message enabled:  0 EN
Destination File Addr:                   N7:0          waiting for queue space:  0 WQ
Target Source File Address:             N7:50
Message Length In Elements:                 10
Message Timeout (seconds):                   5

ERROR CODE: 0                                          control bit address:  N10:0/8
Error Code Desc:
```

- **Timeout Bit TO (word 0, bit 08)** Set this bit in your application to remove an active message instruction from processor control. You can use either your own timeout control routine similiar to the SLC 5/02 MSG instruction or the internal timeout control. We recommend using the built in timeout control because it simplifies the user program.

  To utilize the internal timeout control, a value greater than 0 (typical values are 4 or 5 seconds) must be entered for the MSG instruction time-out parameter. A time-out value of 0 means no time-out value. In other words, if communication is interrupted, the processor will wait forever for a reply. If an ACK is received (as indicated by the ST bit being set), but the reply is not received, the MSG instruction will appear to be locked up, although it is merely waiting for the reply.

  When a value greater than 0 is entered for the MSG time-out parameter and communication is interrupted, the MSG instruction will time-out and error after the time expires, allowing the user program to retry the same message if desired.

Note *When programming timeout control, omit the Timeout Bit manual reset rung. This rung must also be removed from existing (SLC 5/03 OS300) programs when upgrading to new firmware (OS301 or higher).*

- **No Response Bit NR (bit 09)** is set if the target processor responds to the MSG instruction with a target node busy negative acknowledgement for DH-485 protocol only. This means that the target device cannot service the packet at that time and should be retried. The NR bit is reset when the ER or ST bit is set. Do not set or reset this bit. It is informational only.

- **Enabled and Waiting Bit EW (bit 10)** is set after the enable bit is set and the message is buffered and waiting to be sent in the buffer. Do not set or reset this bit. It is informational only.
- **Continuous Operation CO (bit 11)** Set this bit if you wish to continually send the MSG instruction. We recommend that internal timeout control be used for this option and the rung be unconditionally true. Use this bit to turn the mode on and off. A MSG instruction occupies one of the four channel transmission buffers when its CO bit is set. Therefore, a maximum of four MSG instructions per channel may have their CO bit set.

  This mode will continuously operate provided that the rung is continually scanned. If the instruction errors prior to the MSG timeout, it will automatically retry until it is successful. If it times out and is rescanned, the mode will stop. The EN bit must be cleared to resume operation.
- **Error Bit ER (bit 12)** is set when message transmission has failed. The ER bit is reset the next time the associated rung goes from false to true. Do not set or reset this bit. If is informational only.
- **Done Bit DN (bit 13)** is set when the message is transmitted successfully. The DN bit is reset the next time the associated rung goes from false to true. Do not set or reset this bit. It is informational only.
- **Start Bit ST (bit 14)** is set when the processor receives acknowledgement (ACK) from the target device. The ST bit is reset when the DN, ER, or TO bit is set. Do not set or reset this bit. It is informational only.

  For SLC 5/05 Ethernet (channel 1) communications, the ST bit indicates internally that the Ethernet daughterboard has received a command and it is acceptable for a transmission attempt. The command has not yet been transmitted.
- **Enable Bit EN (bit 15)** is set when rung conditions go true and there is space available in either the MSG buffer or MSG queue. It remains set until message transmission is completed and the rung goes false. You may reset this bit once either the ER or DN bit is set in order to retrigger a MSG instruction with true rung conditions on the next scan. Do not set this bit.
- **Waiting for Queue Space Bit WQ (Word 7, bit 0)** is set when the queue is full. This bit is cleared when space is available in the active queue. Do not set or reset this bit. It is informational only.

Note *When the WQ bit is set, or when only the EN bit is set, and you are using a MSG Write instruction, your source data is unbuffered. If your application requires buffered (or "snapshot") data, wait until the EW bit is set before overwriting your source data.*

- EN = 1 and EW = 1 when MSG gets in the buffer
- EN = 1 when MSG gets into queue
- WQ = 1 when queue (which holds 10 MSGs) is full:
  buffer – holds 4 messages with data
  queue – stores pointer (waiting list)

Note *If your program contains four message instructions assigned to the same channel with the Continuous Operation (CO) bit set, no other message instructions can be executed out that same channel, including message instructions which may be in the fault routine.*

The amount of data transferred via a MSG instruction is determined by the size of the destination data type. The limit is 103 words (206 bytes) of data. If a read is used, then the data type in the processor determines the number of elements. If a write, is used then the data type in the remote device determines the number of elements. For example, if a read of counter values from a remote device is done and the destination in the processor is an integer file, then the maximum number of elements that can be requested are 103. The data will come from the first 103 words of the remote counter file.

In contrast, if a read of counter values from a remote device is done and the destination in the processor is a counter file, the maximum number of elements that can be requested is 34, because each counter element contains 3 words.

MSG Instruction Control Block

Limitations for Manipulating the Control Block Bits

Do not manipulate the MSG instruction control block values except as noted below. For example, do not clear the first word of the control block, do not unlatch the time-out control bit, and so on.

The only MSG instruction control bits that may be manipulated by the ladder program without adversely affecting the operation of the instruction are the CO, EN, and TO bits. The enable bit (EN = bit 15) may be unlatched, but only when the done bit (DN = bit 13) or error bit (ER = bit 12) has been set, indicating the successful or unsuccessful completion of the previous message.

In addition, when a MSG is in progress and the ladder program wishes to terminate it for any reason, this may be done by enabling the time-out bit (TO = bit 8). The next time the processor scans the MSG instruction with the TO bit set, it will error the MSG (ER = 1). The MSG instruction may then be re-enabled with a false-to-true transition on the next program scan.

Control Block Layouts

The control block layout is shown below for 500CPU or PLC-5 as the target device:

Read or Write, Local or Remote to a 500CPU or PLC-5

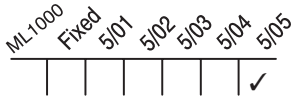
15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	Word
EN ST DN ER CO EW NR TO | Error Code	0
Node Number	1
Reserved for length in words	2
File Number	3
File Type (O, I, S, B, T, C, R, N, F, St, A)	4
Element Number	5
Subelement Number	6
Reserved (Internal Messaging Bits) | WQ	7
Message Timer Preset	8
Message Timer Scaled Zero	9
Message Timer Accumulator	10
Reserved (Internal use only)	11
Reserved (Internal use only)	12
Reserved (Internal use only)	13

The control block layout is shown below for 485CIF as the target device:

Read or Write, Local or Remote to a 485CIF

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	Word
EN ST DN ER CO EW NR TO | Error Code	0
Node Number	1
Reserved for length in words	2
Offset in Words	3
Not Used	4
Not Used	5
Not Used	6
Reserved (Internal Messaging Bits) | WQ	7
Message Timer Preset	8
Message Timer Scaled Zero	9
Message Timer Accumulator	10
Reserved (Internal use only)	11
Reserved (Internal use only)	12
Reserved (Internal use only)	13

SLC 5/05 Message Instruction Overview



Operation

For SLC 5/05 processor Channel 0 communications, the MSG instruction operates the same as it does for the SLC 5/04 (OS401, FRN5 or later) except that the SLC 5/05 also supports logical ASCII addressing for PLC-5 typed read and write messages, which increases the control block size from 14 to 56 words. The SLC 5/05 Channel 1 is dedicated to Ethernet communications and functions as described in the following sections.

Ethernet Connections

TCP/IP is the mechanism used to transport Ethernet messages. On top of TCP, the Client/Server Protocol is required to establish sessions and to send the MSG commands. Connections can be initiated by either a client program (INTERCHANGE or RSLinx application) or a processor.

The client program or processor must first establish a connection to the SLC 5/05 to enable the SLC 5/05 to *receive* solicited messages from a client program or another processor. The client program must also establish a connection to the SLC 5/05 to enable the SLC 5/05 to *send* unsolicited messages to a client program.

In order to send a peer message, the SLC 5/05 must first establish a connection with the destination node at a specified IP address on the Ethernet network. A connection is established when a MSG instruction executes and no previous connection to the same device exists. When a MSG instruction executes, the SLC 5/05 checks to see whether a connection has been established with the destination node. If a connection has not been established, the SLC 5/05 attempts to establish a connection of the peer type. The SLC 5/05 supports a maximum of 16 connections, allowing simultaneous communication with up to 16 other devices or applications. The connections are dedicated as follows:

Number of Connections ①	Dedicated to:
4	peer messages
4	client messages
8	either peer or client messages

① Connections established by an INTERCHANGE client, RSLinx client, and peers are all included when counting the number of connections.

Important: *For peer connections, no more that one connection per destination node is established. If multiple MSG instructions use the same destination node, they share the same connection.*

Ethernet MSG Instruction Parameters

The control block is where all of the information relating to the message is stored. As will be ilustrated, the Ethernet MSG control block length exceeds the normal 14 words used for DH+, DH-485, and serial link MSG instructions.

Attention: While configuring MSG instructions for the DH+ and serial links, keep in mind the files used for Ethernet MSG control blocks.

If you choose a file being used as an Ethernet control block, the programming software prompts you to choose whether you want to overwrite the file. If you choose to overwrite the file, unpredictable machine operation could occur.

After entering the control block, the programming terminal automatically displays a data entry screen, from which you enter instruction parameters that are stored at the control block address.

The table below describes MSG instruction parameters for Ethernet.

Parameter	Value
Supported MSG Commands	485 CIF Read 485 CIF Write PLC5 Typed Read PLC5 Typed Write SLC 500 CPU Read SLC 500 CPU Write
Message Sizes (Channel 1)	256 words maximum, with two exceptions: • PLC5 Type MSG, Timer File – 201 elements maximum • All MSG Types, String File – 23 elements maximum
Modifying Connections	The user may change a MSG instruction destination while the processor is in the RUN mode. If a MSG instruction's destination node changes, the next time the MSG instruction executes, a new connection is established with the new destination node. The old connection will remain open as long as either another MSG instruction was sharing it, or the connection inactive timer has not expired.

MSG Instruction Control Block

Limitations for Manipulating the Control Block Bits

Do not manipulate the MSG instruction control block values except as noted below.. For example, do not clear the first word of the control block, do not unlatch the time-out control bit, and so on.

The only MSG instruction control bits that may be manipulated by the ladder program without adversely affecting the operation of the instruction are the CO, EN, and TO bits. The enable bit (EN = bit 15) may be unlatched, but only when the done bit (DN = bit 13) or error bit (ER = bit 12) has been set, indicating the successful or unsuccessful completion of the previous message.

In addition, when a MSG is in progress and the ladder program wishes to terminate it for any reason, this may be done by enabling the time-out bit (TO = bit 8). The next time the processor scans the MSG instruction with the TO bit set, it will error the MSG (ER = 1). The MSG instruction may then be re-enabled with a false-to-true transition on the next program scan.

Control Block Layouts

The SLC 5/05 MSG control block length varies with the type of communication and with the addressing you use. Control block layouts are shown for:

- SLC 5/05 Channel 0 (RS-232 port)
  MSG Control Block without Logical ASCII Addressing
- SLC 5/05 Channel 0 (RS-232 port)
  MSG Control Block with Logical ASCII Addressing
  valid for PLC-5 typed read or write only
- SLC 5/05 Channel 1 (Ethernet port)
  MSG Control Block without Logical ASCII Addressing
- SLC 5/05 Channel 1 (Ethernet port)
  MSG Control Block with Logical ASCII Addressing
  valid for PLC-5 typed read or write only

The AO bit (word 12, bit 15) is used for PCL-5 type reads and writes. If AO bit is reset to 0, then logical binary addressing is used for PLC-5 type reads and writes. If AO is set to 1, then logical ASCII addressing is selected; in this case the processor expects the ASCII address string information to be stored in words 14 to 55 of the MSG control block (see control block layouts on pages 8–26 and 8–28). The AO bit has no meaning for 485CIF and 500CPU types of reads and writes.

SLC 5/05 Channel 0 (RS-232 port)
MSG Control Block without Logical ASCII Addressing

WORD	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	EN	ST	DN	ER	CO	EW	NR	TO	Error Code							
1	Target Node															
2	Number of Elements															
3	Not Used															
4	File Type (based on **local** source or destination address)															
5	Not Used															
6	Not Used															
7	Reserved (Internal Messaging Bits)															WQ
8	Message Timer Preset															
9	Message Timer Scaled Zero															
10	Message Timer Accumulator															
11	Data Length in Bytes															
12	AO=0	Reserved (Internal Messaging Bits)							Internal Use Only							
13	Reserved															

SLC 5/05 Channel 0 (RS-232 port)
MSG Control Block with Logical ASCII Addressing
valid for PLC-5 typed read or write only

WORD	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
0	EN	ST	DN	ER	CO	EW	NR	TO	Error Code							
1	Target Node															
2	Number of Elements															
3	Not Used															
4	File Type (based on **local** source or destination address)															
5	Not Used															
6	Not Used															
7	Reserved (Internal Messaging Bits)															WQ
8	Message Timer Preset															
9	Message Timer Scaled Zero															
10	Message Timer Accumulator															
11	Data Length in Bytes															
12	AO=1	Reserved (Internal Messaging Bits)							Internal Use Only							
13	Reserved															
14	Logical ASCII Address String Length including NULL Termination Character (bytes)															
15	First Byte of Address String								Second Byte of Address String							
16	Third Byte of Address String								…							
	…								…							
	…								…							
55	Eighty-First Byte of ASCII Address String								NULL Byte of Longest ASCII Address String							

	SLC 5/05 Channel 1 (Ethernet port) MSG Control Block without Logical ASCII Addressing															
WORD	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
0	EN	ST	DN	ER	CO	EW	NR	TO	Error Code							
1	Reserved (Target Node Not Used)															
2	Number of Elements															
3	Not Used															
4	File Type (based on **local** source or destination address)															
5	Not Used															
6	Not Used															
7	Reserved (Internal Messaging Bits)															WQ
8	Message Timer Preset															
9	Message Timer Scaled Zero															
10	Message Timer Accumulator															
11	Data Length in Bytes															
12	AO=0	Reserved (Internal Messaging Bits)							Reserved							
13	Reserved															
14	First Byte of IP Address String①								Second Byte of IP Address String							
15	Third Byte of IP Address String								…							
	…								…							
	…								…							
34	Forty-First Byte of IP Address String								NULL Byte of Longest IP Address String							
35	Reserved								Reserved (Ethernet Message Type); must be 0							
36–50	Reserved for Future Use															

① The IP Address string format is up to 42 ASCII characters including a terminating NULL character. The first byte in the array is the left most character in the string as written. For example: If the IP Address is 423.156.78.012, the first byte is the ASCII character “4”. If the MSG destination is an INTERCHANGE client on a host computer, the destination is specified as “client” and stored as a NULL terminated string.

	SLC 5/05 Channel 1 (Ethernet port) MSG Control Block with Logical ASCII Addressing *valid for PLC-5 typed read or write only*															
WORD	**15**	**14**	**13**	**12**	**11**	**10**	**09**	**08**	**07**	**06**	**05**	**04**	**03**	**02**	**01**	**00**
0	EN	ST	DN	ER	CO	EW	NR	TO	Error Code							
1	Reserved (Target Node Not Used)															
2	Number of Elements															
3	Not Used															
4	File Type (based on **local** source or destination address)															
5	Not Used															
6	Not Used															
7	Reserved (Internal Messaging Bits)															WQ
8	Message Timer Preset															
9	Message Timer Scaled Zero															
10	Message Timer Accumulator															
11	Data Length in Bytes															
12	AO=1	Reserved (Internal Messaging Bits)							Reserved							
13	Reserved															
14	Logical ASCII Address String Length including NULL Termination Character (bytes)															
15	First Byte of ASCII Address String								Second Byte of ASCII Address String							
16	Third Byte of ASCII Address String								…							
	…								…							
	…								…							
55	Eighty-First Byte of ASCII Address String								NULL Byte of Longest ASCII Address String							
56	First Byte of IP Address String①								Second Byte of IP Address String							
57	Third Byte of IP Address String								…							
	…								…							
	…								…							
76	Forty-First Byte of IP Address String								NULL Byte of Longest IP Address String							
77	Reserved								Reserved (Ethernet Message Type); must be 0							
78–92	Reserved for Future Use															

① The IP Address string format is up to 42 ASCII characters including a terminating NULL character. The first byte in the array is the left most character in the string as written. For example: If the IP Address is 423.156.78.012, the first byte is the ASCII character "4". If the MSG destination is an INTERCHANGE client on a host computer, the destination is specified as "client" and stored as a NULL terminated string.

Interpreting Ethernet Status Data

Monitor the status of SLC 5/05 processors by accessing the Ethernet channel 1 status screen of your programming software.

```
                              Ethernet
                        Channel 1 Diagnostic Counters
Commands
       sent:                          received:
Replies
       sent:                          received
        sent with error:               received with error:
       timed out:
Ethernet
       In Octets:                      Out Octets:
       In Packets:                     Out Packets:
        alignment errors:              FCS errors:
        carrier sense errors           excessive collisions:
        excessive deferrals            MAC receive errors:
        MAC transmit errors            single collisions
        multiple collisions            deferred transmission:
        late collisions
```

The diagnostic counter data displayed is stored in the diagnostic file defined on the Ethernet channel 1 configuration screen.

Status field:		Bytes:	Displays the number of:
Commands	sent	0-3	Commands sent by the channel
	received	4-7	Commands received by the channel
Replies	sent	8-11	Replies sent by the channel
	received	12-15	Replies received by the channel
	sent with error	16-19	Replies containing errors sent by the channel
	received with error	20-23	Replies containing errors received by the channel
	timed out	24-27	Replies not received within the specified timeout period
Ethernet	In Octets	28-31	Octets received on the channel
	Out Octets	32-35	Octets sent on the channel
	In Packets	36-39	Packets received on the channel, including broadcast packets
	Out Packets	40-43	Packets sent on the channel, including broadcast packets
	alignment errors	44-47	Frames received on the channel that are not an integral number of octets in length
	FCS errors	48-51	Frames received on the channel that do not pass the FCS check
	carrier sense errors	52-55	Times that the carrier sense condition was lost or never asserted while trying to transmit a frame
	excessive collisions	56-59	Frames for which a transmission fails due to excessive collisions
	excessive deferrals	60-63	Frames for which transmission is deferred for an excessive period of time
	MAC receive errors	64-67	Frames for which reception on an interface fails due to internal MAC sublayer receive error
	MAC transmit errors	68-71	Frames for which reception on an interface fails due to internal MAC sublayer transmit error
	single collisions	72-75	Successfully transmitted frames for which transmission was delayed because of collision.
	multiple collisions	76-79	Successfully transmitted frames for which transmission was delayed more than once because of collision.
	deferred transmission	80-83	Frames for which the first transmission attempt is delayed because the medium is busy
	late collisions	84-87	Times that a collision is detected later than 512 bit-times into the transmission of a packet

MSG Instruction Error Codes

When the processor detects an error during the transfer of message data, the processor sets the .ER bit and enters an error code that you can monitor from your programming software.

Error Code	Description of Error Condition
02H	Target node is busy. The MSG instruction will automatically reload. If other messages are waiting, the message is placed at the bottom of the stack.
03H	Target node cannot respond because message is too large.
04H	Target node cannot respond because it does not understand the command parameters OR the control block may have been inadvertently modified.
05H	Local processor is offline (possible duplicate node situation).
06H	Target node cannot respond because requested function is not available.
07H	Target node does not respond.
08H	Target node cannot respond.
09H	Local modem connection has been lost.
0AH	Buffer unavailable to receive SRD reply.
0BH	Target node does not accept this type of MSG instruction.
0CH	Received a master link reset (one possible source is from the DF1 master).
10H	Target node cannot respond because of incorrect command parameters or unsupported command.
11H	Local file has constant file protection.
12H	Local channel configuration protocol error exists.
13H	Local MSG configuration error in the Remote MSG parameters.
15H	Local channel configuration parameter error exists.
16H	Target or Local Bridge address is higher than the maximum node address.
17H	Local service is not supported.
18H	Broadcast (Node Address 255) is not supported.
19H	Improperly formatted Logical ASCII Address string. String not properly terminated with a NULL character or the string length does not match the value in the length parameter.
20H	PCCC Description: Host has a problem and will not communicate.
30H	PCCC Description: Remote station host is not there, disconnected, or shutdown.
37H	Message timed out in local processor.
38H	Message disabled pending link response.
40H	PCCC Description: Host could not complete function due to hardware fault.
50H	Target node is out of memory.
60H	Target node cannot respond because file is protected.
70H	PCCC Description: Processor is in Program Mode.

Error Code	Description of Error Condition
80H	PCCC Description: Compatibility mode file missing or communication zone problem.
90H	PCCC Description: Remote station cannot buffer command.
B0H	PCCC Description: Remote station problem due to download.
C0H	PCCC Description: Cannot execute command due to active IPBs.
D0H	No IP address configured for the network, -or- Bad command – unsolicited message error, -or- Bad address – unsolicited message error, -or- No privilege – unsolicited message error
D1H	Maximum connections used – no connections available
D2H	Invalid internet address or host name
D3H	No such host / Cannot communicate with the name server
D4H	Connection not completed before user-specified timeout
D5H	Connection timed out by the network
D7H	Connection refused by destination host
D8H	Connection was broken
D9H	Reply not received before user-specified timeout
DAH	No network buffer space available
E1H	PCCC Description: Illegal Address Format, a field has an illegal value.
E2H	PCCC Description: Illegal Address format, not enough fields specified.
E3H	PCCC Description: Illegal Address format, too many fields specified.
E4H	PCCC Description: Illegal Address, symbol not found.
E5H	PCCC Description: Illegal Address Format, symbol is 0 or greater than the maximum number of characters support by this device.
E6H	PCCC Description: Illegal Address, address does not exist, or does not point to something usable by this command.
E7H	Target node cannot respond because length requested is too large.
E8H	PCCC Description: Cannot complete request, situation changed (file size, for example) during multi-packet operation.
E9H	PCCC Description: Data or file is too large. Memory unavailable.
EAH	PCCC Description: Request is too large; transaction size plus word address is too large.
EBH	Target node cannot respond because target node denies access.
ECH	Target node cannot respond because requested function is currently unavailable.
EDH	PCCC Description: Resource is already available; condition already exists.
EEH	PCCC Description: Command cannot be executed.
EFH	PCCC Description: Overflow; histogram overflow.
F0H	PCCC Description: No access

Error Code	Description of Error Condition
F1H	Local processor detects illegal target file type.
F2H	PCCC Description: Invalid parameter; invalid data in search or command block.
F3H	PCCC Description: Address reference exists to deleted area.
F4H	PCCC Description: Command execution failure for unknown reason; PLC-3 histogram overflow.
F5H	PCCC Description: Data conversion error.
F6H	PCCC Description: The scanner is not able to communicate with a 1771 rack adapter. This could be due to the scanner not scanning, the selected adapter not being scanned, the adapter not responding, or an invalid request of a “DCM BT (block transfer)”.
F7H	PCCC Description: The adapter is not able to communicate with a module.
F8H	PCCC Description: The 1771 module response was not valid – size, checksum, etc.
F9H	PCCC Description: Duplicated Label.
FAH	Target node cannot respond because another node is file owner (has sole file access).
FBH	Target node cannot respond because another node is program owner (has sole access to all files).
FCH	PCCC Description: Disk file is write protected or otherwise inaccessible (off-line only).
FDH	PCCC Description: Disk file is being used by another application; update not performed (off-line only).
FFH	Local communication channel is shut down.

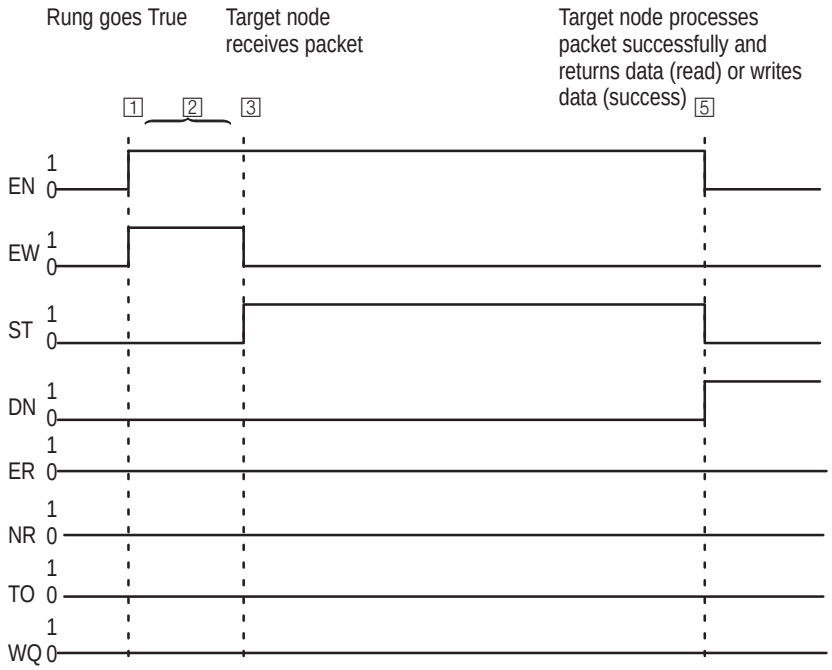
Note

For 1770–6.5.16 DF1 Protocol and Command Set Reference Manual users:

The MSG error code reflects the STS field of the reply to your MSG instruction.
Codes E0 – EF represent EXT STS codes 0 – F.
Codes F0 – FC represent EXT STS codes 10 – 1C.

Timing Diagram for SLC 5/03, SLC 5/04, and SLC 5/05 MSG Instruction

The following section describes the timing diagram for a SLC 5/03, SLC 5/04, or SLC 5/05 MSG instruction.



1. When the MSG rung becomes true and the MSG is scanned, if there is room in any of the four active MSG buffers, the **EN** and **EW** bits are set. If this were a MSG Write instruction, the source data would be transferred to the MSG buffer at this time. If there is no room in the four MSG buffers, but a position is available in the 10-position MSG Queue, only the **EN** bit is set. The 10-position MSG Queue works on a first-in-first-out basis that allows the SLC processor to remember the order the MSG instructions were enabled. Note that the program does not have access to the SLC MSG Queue.

If there is no room in any of the four MSG buffers and no room in the 10-position MSG Queue, only the **WQ** bit is set. Note that when the **WQ** bit is set, the MSG instruction must be re-scanned with true rung conditions at a later time when there is room in either the four MSG buffers or the 10-position MSG Queue.

Once the **EN** bit is set, it remains set until the entire MSG process is complete and either the **DN**, **ER**, or **TO** bit is set. The **MSG Timeout** period begins timing when the **EN** bit is set. If the timeout period expires before the MSG instruction completes it function, the **ER** bit is set and an error code (37H) is placed in the MSG block to inform you of the timeout error.

If you choose to set the **CO** bit, your MSG instruction will "take up" permanent residence in one of the four active MSG buffers. The MSG instruction continues to re-transmit its data each time the **DN** or **ER** bit is set. If this were a MSG Write instruction, your source data would be updated each MSG cycle.

2. At the next end of scan or SVC, the SLC processor determines if it should examine the MSG Queue for "something to do." The processor bases its decision on the state of bits S:2/15, S:33/7, S:33/5, S:33/6, network communication requests from other nodes, and if previous MSG instructions are already in progress. If the SLC processor determines that it should not access the queue, the MSG instruction remains as it was. (Either the **EN** and **EW** bits remain set, or only the **EN** bit is set, or only the **WQ** bit is set until the next end of scan or SVC. If only the **WQ** bit is set, the MSG instruction must be re-scanned later with true rung conditions.)

If the SLC processor determines that it has "something to do," it unloads the MSG Queue entries into the MSG buffers until all four MSG buffers are full. Each MSG buffer contains a valid network packet. If a packet cannot be successfully built from the MSG Queue, the **ER** bit is set and a code is placed in the MSG block to inform you of an error. When a MSG instruction is loaded into a MSG buffer, the **EN** and **EW** bits are set.

The SLC processor then exits the end of scan or SVC portion of the scan. The processor's background communication function sends the packets to the Target Nodes that you specified in your MSG instruction. Depending on the state of bits S:2/15, S:33/7, S:33/5, and S:33/6 you can have up to four MSG instructions active at any given time (eight MSG instructions for SLC 5/03 OS301 or higher, SLC 5/04, and SLC 5/05 processors).

3. If the Target Node successfully receives the packet, it sends back an ACK (acknowledge). The ACK causes the processor to clear the **EW** bit and set the **ST** bit. The Target Node has not yet examined the packet, to see if it understands your request. Note that the Target Node is not required to respond within any give time frame.

For SLC 5/05 Ethernet communication, there is no ACK/NAK mechanism. The ST bit is set when the Ethernet daughterboard internally indicates it has received the command from the main processor and will send it out. Skip step 4 for SLC 5/05 processors.

Note *If the Target Node faults or power cycles during this time frame of a MSG transaction, you will never receive a reply. This is why it is recommended to use a **MSG Timeout** value in your MSG instruction.*

Step 4 not shown in the timing diagram.

4. If you do not receive an ACK, step 3 does not occur. Instead either no response a NAK (no acknowledge) is received. When this happens, the **ST** bit remains clear.

No response may be caused by:

- the target node is not there
- the target node does not respond because the packet became too corrupted in tranmission to be properly received
- the response was corrupted in transmission back

A NAK may be caused by:

- target node is too busy
- target node received a corrupt packet.

When a NAK occurs, the **EW** bit is cleared and the **NR** bit is set for one scan. The next time the MSG instruction is scanned, the **ER** bit is set and the **NR** bit is cleared. This indicates that the MSG instruction failed. Note that if the Target Node is too busy, the **ER** bit is not set. Instead, the MSG instruction re-queues itself for re-transmission.

5. Following the successful receipt of the packet, the Target Node sends a reply packet. The reply packet will contain one of the following responses:

- I have successfully performed your write request.
- I have successfully performed your read request, and here is your data.
- I have not performed your request, you are in error.

At the next end of scan or SVC, following the Target Node's reply, the SLC processor examines the packet from the target device. If the reply contains "I have successfully performed your write request," the **DN** bit is set and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

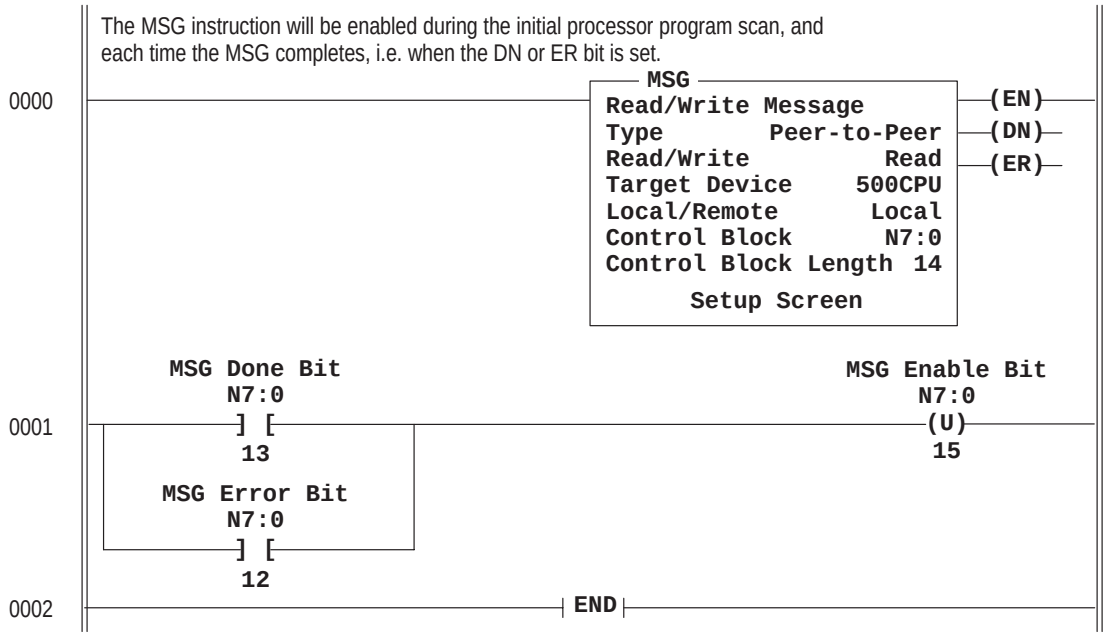
If the reply contains "I have successfully performed your read request, and here is your data," the data is written to the data table, the **DN** bit is set and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

If the reply contains “I have not performed your request, you are in error,” the **ER** bit is set and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

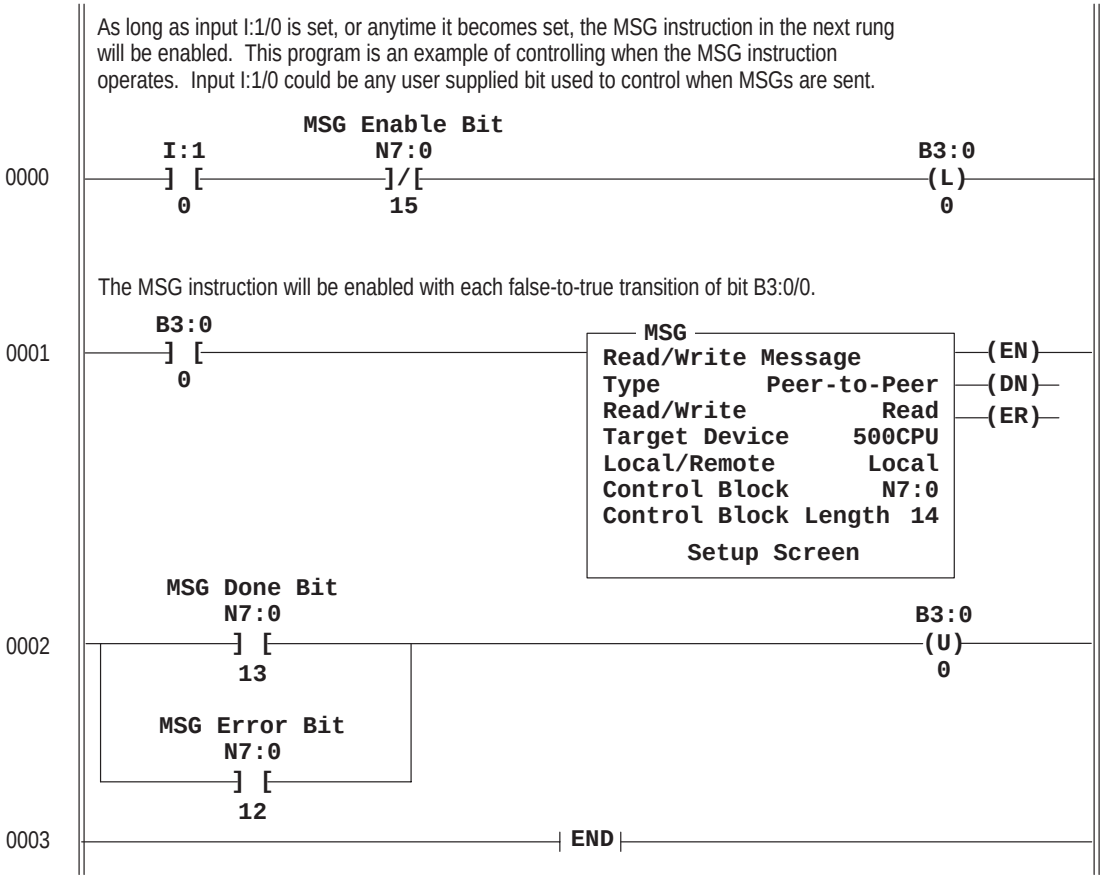
The four MSG buffers are shared between channel 0 and channel 1 for SLC 5/03 OS300 processors. For SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors, there are four MSG buffers per channel. Each channel has its own 10-position MSG Queue. The SLC processor unloads the two MSG queues into the MSG buffers evenly at end of scan or SVC. This allows both channels equal access to communications. If you program a SVC instruction that is configured to service only one channel, then only that channel will have its MSG Queue unloaded into the MSG buffers (until the next end of scan or SVC when both channels will again be unloaded evenly).

Examples: Ladder Logic

Enabling the MSG Instruction Via Ladder Logic



Enabling the MSG Instruction Via User Supplied Input



Using Local Messaging

Example 1 – Local Read from a 500CPU

```
Type:                          Peer-to-Peer
Read/Write:                          READ                ignore if timed out:  0 TO
Target Device:                      500CPU                      to be retried:  0 NR
Local/Remote:                        Local               awaiting execution:  0 EW
Control Block:                       N10:0                   continuous run:  0 CO
Channel:                                0                            error:  0 ER
Target Node (decimal):                  2                     message done:  0 DN
                                                          message transmitting:  0 ST
                                                              message enabled:  0 EN
Destination File Addr:               N7:0               waiting for queue space:  0 WQ
Target Source File Address:         N7:50
Message Length In Elements:            10
Message Timeout (seconds):              5

ERROR CODE: 0                                          control bit address:  N10:0/8
Error Code Desc:
```

In the display above the SLC 5/03 or SLC 5/04 processor reads 10 elements from Target Node 2's N7 file, starting at word N7:50. The 10 words are placed in your integer file starting at word N7:0. If five seconds elapse without a reply, error bit N10:0/12 is set, indicating that the instruction timed out. The device at node 2 understands the SLC 500 processor family (SLC 500, SLC 5/01, SLC 5/02, SLC 5/03, SLC 5/04, SLC 5/05 and MicroLogix 1000) protocol.

Function Key	Description
Target Node	Specifies the node number of the processor that is receiving the message. Valid range is 0–31 for DH–485 protocol, or 0–254 for DF1 protocol.
File Address	For a Read (Destination) this is the address in the initiating processor which is to receive data.
	For a Write (Source) this is the address in the initiating processor which is to send data.
	Valid file types are S, B, T, C, R, N, I, O, M0, M1, F, ST, and A.
Target Address	For a Read (Source) this is the address in the target processor which is to send data.
	For a Write (Destination) this is the address in the target processor which is to receive data.
	Valid file types are S, B, T, C, R, N, I, O, M0, M1, F, ST and A.
Message Length	Defines the length of the message in elements. One word elements are limited to a maximum length of 1–103. Three word elements are limited to a maximum length of 1–37.
Message Timeout	Defines the length of the message timer in seconds. A timeout of 0 seconds means that there is no timer and the message will wait indefinitely for a reply. Valid range is 0–255 seconds.
Channel	Identifies the physical channel used for the message communication. Available channels: SLC 5/03 – (Channel 0, RS–232) or (Channel 1, DH–485) SLC 5/04 – (Channel 0, RS–232) or (Channel 1, DH+) SLC1 5/05 – (Channel 0, RS-232) or (Channel 1, Ethernet)

Example 2 – Local Read from a 485CIF

```
Type:                          Peer-to-Peer
Read/Write:                            READ              ignore if timed out:  0 TO
Target Device:                       485CIF                    to be retried:  0 NR
Local/Remote:                         Local             awaiting execution:  0 EW
Control Block:                        N10:0                 continuous run:  0 CO
Channel:                                  0                          error:  0 ER
Target Node (decimal):                    2                   message done:  0 DN
                                                          message transmitting:  0 ST
                                                              message enabled:  0 EN
Destination File Addr:                 N7:0          waiting for queue space:  0 WQ
Target Offset:                           20
Message Length In Elements:               5
Message Timeout (seconds):               15

ERROR CODE: 0                                          control bit address:  N10:0/8
Error Code Desc:
```

In the display above the SLC processor reads five elements (words) from Target Node 2's CIF file, starting at word 20 (or byte 20 for non–SLC 500 devices). The five elements are placed in your integer file starting at word N7:0. If 15 seconds elapse without a reply, error bit N10:0/12 is set, indicating that the instruction timed out. The device at node 2 understands the 485CIF (PLC–2 emulation) protocol.

Function Key	Description
Target Node	Specifies the node number of the processor that is receiving the message. Valid range is 0–31 for DH-485 protocol, or 0–254 for DF1 protocol.
File Address	For a Read (Destination) this is the address in the initiating processor which is to receive data.
	For a Write (Source) this is the address in the initiating processor which is to send data.
	Valid file types are S, B, T, C, R, N, I, O, M0, M1, F, ST, and A.
Target Offset	For a Read or Write this is the word offset value in the common interface file (byte offset for non–SLC device).
Message Length	When using a 485CIF Message instruction, the message length is the number of 16-bit words. You can specify 1 to 103 elements (words of information).
Message Timeout	Defines the length of the message timer in seconds. A timeout of 0 seconds means that there is no timer and the message will wait forever for a reply. Valid range is 0–255 seconds.
Channel	Identifies the physical channel used for the message communication. Available channels: SLC 5/03 – (Channel 0, RS–232) or (Channel 1, DH–485) SLC 5/04 – (Channel 0, RS–232) or (Channel 1, DH+) SLC 5/05 – (Channel 0, RS-232) or (Channel 1, Ethernet)

Example 3 – Local Read from a PLC-5

```
Type:                          Peer-to-Peer
Read/Write:                            READ              ignore if timed out:  0 TO
Target Device:                         PLC5                    to be retried:  0 NR
Local/Remote:                         Local             awaiting execution:  0 EW
Control Block:                        N10:0                 continuous run:  0 CO
Channel:                                  0                          error:  0 ER
Target Node (decimal):                    2                   message done:  0 DN
                                                         message transmitting:  0 ST
                                                              message enabled:  0 EN
Destination File Addr:                 N7:0          waiting for queue space:  0 WQ
Target Src/Dst File Address:          N7:50
Message Length In Elements:              10
Message Timeout (seconds):                5

ERROR CODE:  0                                        control bit address:  N10:0/8
Error Code Desc:
```

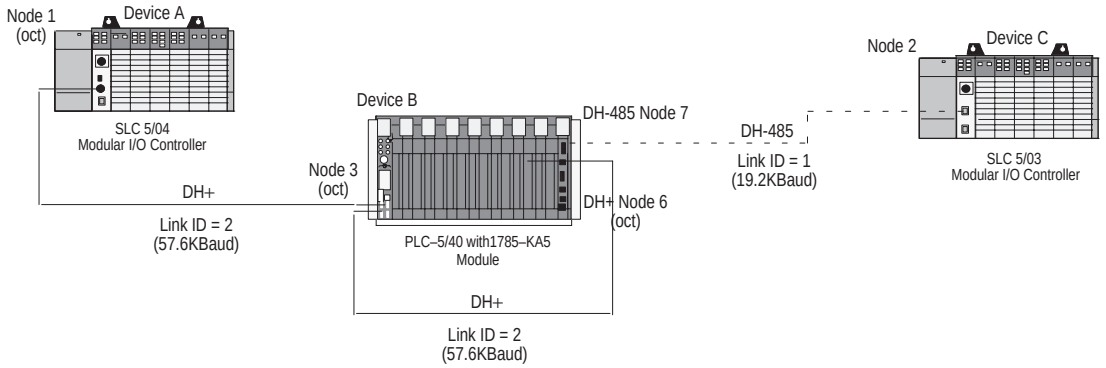
In the display above the SLC 5/03 or SLC 5/04 processor reads 10 elements from Target Node 2's N7 file, starting at word N7:50. The 10 words are placed in your integer file starting at word N7:0. If five seconds elapse without a reply, error bit N10:0/12 is set, indicating that the instruction timed out. The device at node 2 understands the PLC-5 processor protocol.

Function Key	Description
Target Node	Specifies the node number of the processor that is receiving the message. Valid range is 0–31 for DH-485 protocol, or 0–254 for DF1 protocol.
File Address	For a Read (Destination) this is the address in the initiating processor which is to receive data.
	For a Write (Source) this is the address in the initiating processor which is to send data.
	Valid file types are S, B, T, C, R, N, I, O, F, ST, and A.
Target Address	For a Read (Source) this is the address in the target processor which is to send data.
	For a Write (Destination) this is the address in the target processor which is to receive data.
	Valid file types are S, B, T, C, R, N, I, O, F, ST, and A.
Message Length	Defines the length of the message in elements. One word elements are limited to a maximum length of 1–103. Three word elements are limited to a maximum length of 1–37.
Message Timeout	Defines the length of the message timer in seconds. A timeout of 0 seconds means that there is no timer and the message will wait indefinitely for a reply. Valid range is 0–255 seconds.
Channel	Identifies the physical channel used for the message communication. Available channels: SLC 5/03 – (Channel 0, RS–232) or (Channel 1, DH–485) SLC 5/04 – (Channel 0, RS–232) or (Channel 1, DH+) SLC 5/05 – (Channel 0, RS-232) or (Channel 1, Ethernet)

Using Remote Messaging

The SLC 5/03 and higher processors can pass a MSG instruction through one remote network to its target destination. (You can make one hop across a network.) The SLC 5/03 and higher processors can also pass a MSG instruction to the network that exists on the other side of the local bridge.

Example 1 – Communicating with A–B processors using a 1785-KA5



SLC 5/04 Processor (A) to SLC 5/03 Processor (C) via 1785-KA5

```
Type:                                Peer–to–Peer
Read/Write:                                Read
Target Device:                          500 CPU
Local/Remote:                            Remote
Control Block:                     user specified
Channel:                                      1
Target Node (decimal):                        2
Remote Bridge Link Id <dec>:                  1
Remote Bridge Node Address <dec>:             0
Local Bridge Node Address <dec>:              6
Destination/Source File Addr:      user specified
Target Src/Dst File Address:       user specified
Message Length In Elements:        user specified
Message Timeout (seconds):         user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/04 processor on the DH+ (Link ID 2).

Target Node is the SLC 5/03 processor at node address 2.

Remote Bridge Link ID is the link ID of the remote DH-485 network with the 1785-KA5 and SLC 5/03 processor (Link ID 1).

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 6 since this is the DH+ node address used by the 1785-KA5 communication interface module.

SLC 5/03 Processor (C) to SLC 5/04 Processor (A) via 1785-KA5

```
Type:                                Peer–to–Peer
Read/Write:                             Read
Target Device:                       500 CPU
Local/Remote:                         Remote
Control Block:                    user specified
Channel:                                 1
Target Node (decimal):                   1
Remote Bridge Link Id <dec>:             2
Remote Bridge Node Address <dec>:        0
Local Bridge Node Address <dec>:         7
Destination/Source File Addr:     user specified
Target Src/Dst File Address:      user specified
Message Length In Elements:       user specified
Message Timeout (seconds):        user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/03 processor on the DH-485 (Link ID 1).

Target Node is the SLC 5/04 processor at node address 1.

Remote Bridge Link ID is the link ID of the remote DH+ network with the 1785-KA5 and the SLC 5/04 processor (Link ID 2).

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 7 since this is the DH-485 node address used by the 1785-KA5 communication interface module.

SLC 5/03 Processor (C) to a PLC-5 (B) via 1785-KA5

```
Type:                                Peer–to–Peer
Read/Write:                             Write
Target Device:                          PLC5
Local/Remote:                         Remote
Control Block:                  user specified
Channel:                                 1
Target Node (decimal):                   3
Remote Bridge Link Id <dec>:             2
Remote Bridge Node Address <dec>:        0
Local Bridge Node Address <dec>:         7
Destination/Source File Addr:   user specified
Target Src/Dst File Address:    user specified
Message Length In Elements:     user specified
Message Timeout (seconds):      user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/03 processor on the DH-485 (Link ID 1).

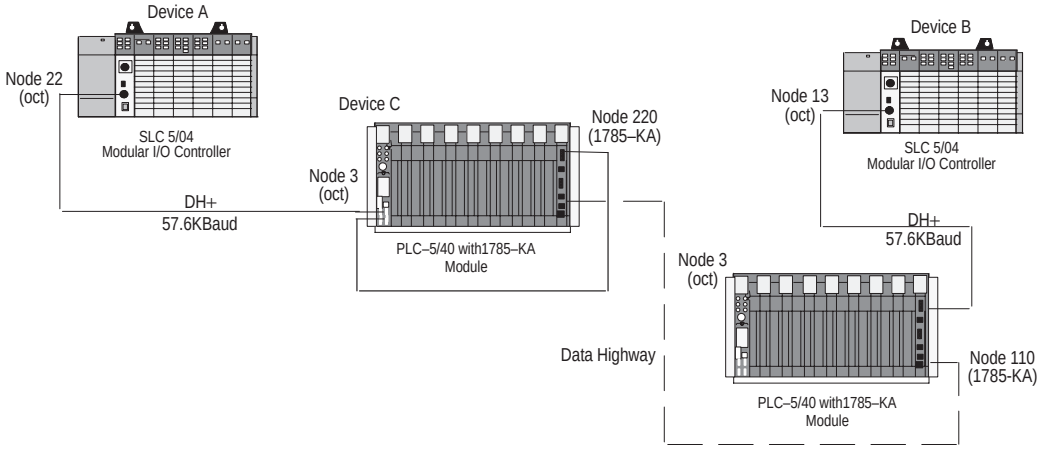
Target Node is the PLC-5 processor at node address 3.

Remote Bridge Link ID is the link ID of the remote DH+ network with the 1785-KA5 and the PLC-5 processor (Link ID 2).

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 7 since this is the DH-485 node address used by the 1785-KA5 communication interface module.

Example 2 – Communicating with A–B processors using two 1785-KAs



SLC 5/04 Processor (B) to PLC5 Processor (C) via two 1785–KAs

```
Type:                                Peer–to–Peer
Read/Write:                             Write
Target Device:                           PLC5
Local/Remote:                          Remote
Control Block:                   user specified
Channel:                                    1
Target Node (decimal):                      0
Remote Bridge Link Id <dec>:                0
Remote Bridge Node Address <dec>:         131
Local Bridge Node Address <dec>:            8
Destination/Source File Addr:    user specified
Target Src/Dst File Address:     user specified
Message Length In Elements:      user specified
Message Timeout (seconds):       user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/04 processor on the DH+ network.

Target Node is the PLC-5 processor at node address 0. (This is actually node address 3, but the node address is set to 0 because the Remote Bridge Node address handles the addressing structure.)

Remote Bridge Link ID is set always set to 0 when using this addressing structure.

Remote Bridge Node Address is set to 131. The remote bridge node address consists of the most significant digit (octal) of the remote 1785-KA (220) plus the target node address. For example, 200 + 3 = 203 octal (131 decimal).

Local Bridge Node Address is set to 8 since this is the decimal equivalent of the two least significant digits of the 1785-KA address (10 octal).

SLC 5/04 Processor (B) to SLC 5/04 Processor (A) via two 1785–KAs

```
Type:                                Peer–to–Peer
Read/Write:                            Write
Target Device:                       500 CPU
Local/Remote:                         Remote
Control Block:                    user specified
Channel:                                 1
Target Node (decimal):                   0
Remote Bridge Link Id <dec>:             0
Remote Bridge Node Address <dec>:      146
Local Bridge Node Address <dec>:         8
Destination/Source File Addr:     user specified
Target Src/Dst File Address:      user specified
Message Length In Elements:       user specified
Message Timeout (seconds):        user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/04 processor on the DH+ network.

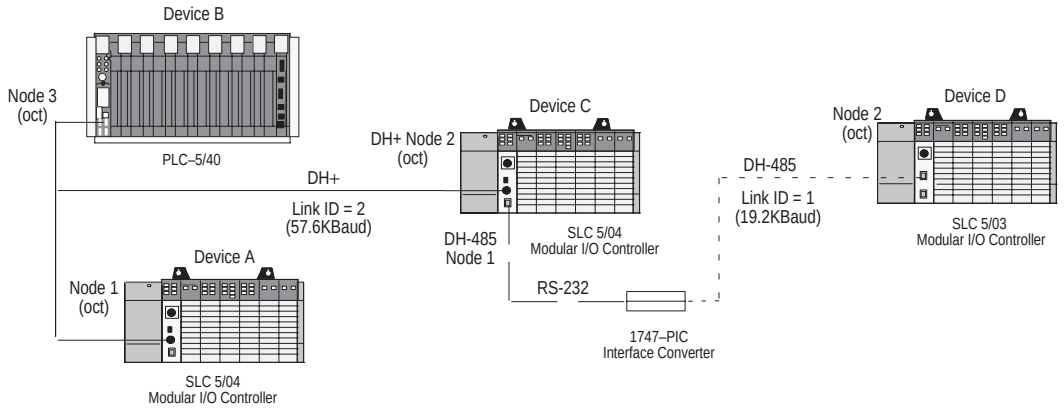
Target Node is the SLC 5/04 processor at node address 0. (This is actually node address 22, but the node address is set to 0 because the Remote Bridge Node address handles the addressing structure.)

Remote Bridge Link ID is always set to 0 when using this addressing structure.

Remote Bridge Node Address is set to 146. The remote bridge node address consists of the most significant digit (octal) of the remote 1785-KA plus the address of the target device. For example, 200 + 22 = 222 (146 decimal).

Local Bridge Node Address is set to 8 since this is the decimal equivalent of the two least significant digits of the 1785-KA address (10 octal).

Example 3 – Passthru via DH-485 Channel 0 of the SLC 5/04 Processor



SLC 5/04 Processor (A) to SLC 5/03 Processor (D) via an SLC 5/04 Processor (C) (Passthru using Channel 0 DH-485)

```
Type:                                Peer–to–Peer
Read/Write:                                Read
Target Device:                          500 CPU
Local/Remote:                            Remote
Control Block:                    user specified
Channel:                                      1
Target Node (decimal):                        2
Remote Bridge Link Id <dec>:                  1
Remote Bridge Node Address <dec>:             0
Local Bridge Node Address <dec>:              2
Destination/Source File Addr:     user specified
Target Src/Dst File Address:      user specified
Message Length In Elements:       user specified
Message Timeout (seconds):        user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/04 processor on the DH+ network.

Target Node is the SLC 5/03 processor at node address 2.

Remote Bridge Link ID is the link ID of the remote DH-485 network with the SLC 5/04 processor (Channel 0, Link ID 1) and the SLC 5/03 processor (Channel 0, Link ID 1).

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 2 since this is the DH+ node address used by the passthru SLC 5/04 processor.

SLC 5/03 Processor (D) to SLC 5/04 Processor (A) via an SLC 5/04 Processor (C) (Passthru using Channel 0 DH–485)

```
Type:                                Peer–to–Peer
Read/Write:                                Read
Target Device:                          500 CPU
Local/Remote:                            Remote
Control Block:                     user specified
Channel:                                      1
Target Node (decimal):                        3
Remote Bridge Link Id <dec>:                  2
Remote Bridge Node Address <dec>:             0
Local Bridge Node Address <dec>:              1
Destination/Source File Addr:      user specified
Target Src/Dst File Address:       user specified
Message Length In Elements:        user specified
Message Timeout (seconds):         user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/03 processor on the DH-485 network.

Target Node is the SLC 5/04 processor at node address 1.

Remote Bridge Link ID is the link ID of the remote DH+ network with both SLC 5/04 processors (Channel 1, Link ID 2).

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 1 since this is the DH-485 node address used by the passthru SLC 5/04 processor.

SLC 5/03 Processor (D) to PLC-5 (B) via an SLC 5/04 Processor (Passthru using Channel 0 DH-485)

```
Type:                                Peer–to–Peer
Read/Write:                             Write
Target Device:                          PLC5
Local/Remote:                         Remote
Control Block:                  user specified
Channel:                                 1
Target Node (decimal):                   3
Remote Bridge Link Id <dec>:             2
Remote Bridge Node Address <dec>:        0
Local Bridge Node Address <dec>:         1
Destination/Source File Addr:   user specified
Target Src/Dst File Address:    user specified
Message Length In Elements:     user specified
Message Timeout (seconds):      user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/03 processor on the DH-485 network.

Target Node is the PLC-5 processor at node address 3.

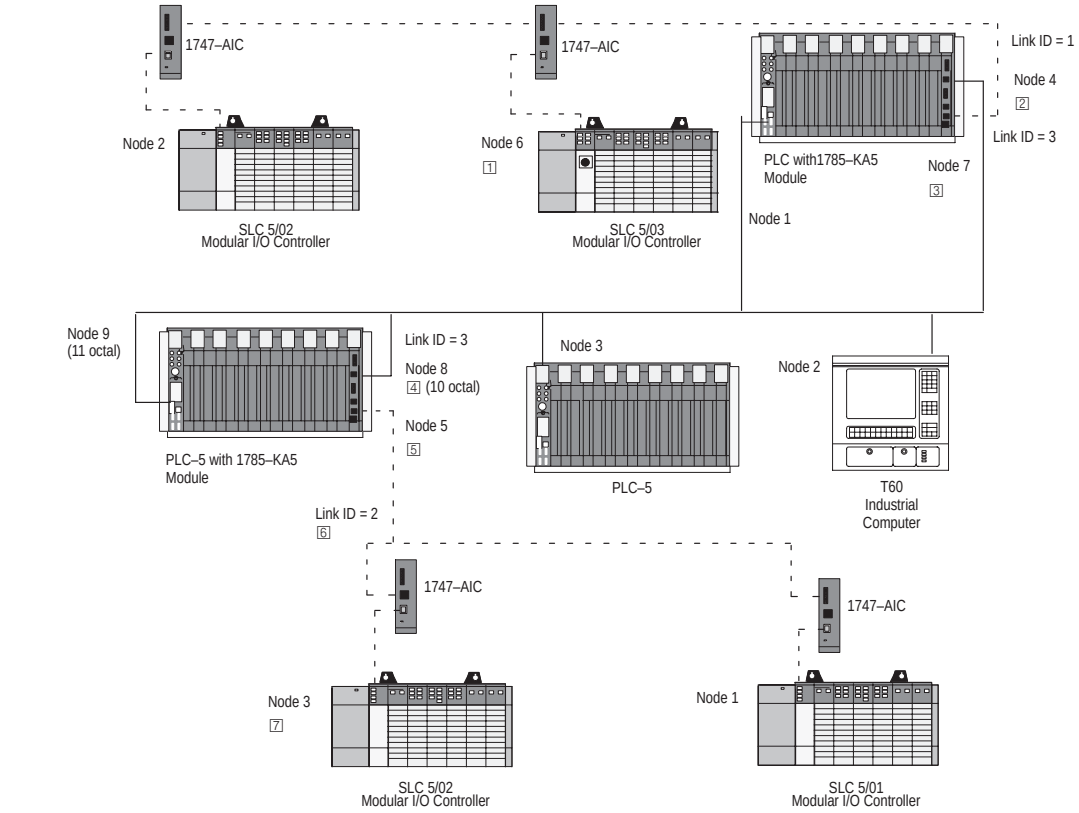
Remote Bridge Link ID is the link ID of the remote DH+ network with the SLC 5/04 processor (Channel 1, Link ID 2) and PLC-5 processor (Channel 1A, Link ID 2).

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 1 since this is the DH-485 node address used by the passthru SLC 5/04 processor.

Remote Messaging (SLC 5/03 to a SLC 500, SLC 5/01, or SLC 5/02, or MicroLogix 1000)

The following illustration shows the connectivity for a remote message.

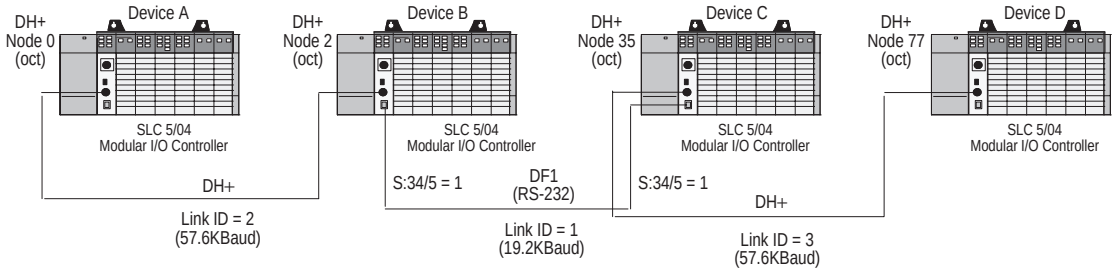


The following callouts depict addressing parameters of a SLC 5/03 to a remote SLC 5/02 processor.

1. This is the orginating node of the MSG instruction. You do not need to specify its address.
2. This is the Local Bridge Node Address.
3. This is the remote node address of the local bridge. You do not need to specify its address.
4. This is the Remote Bridge Node Address.
5. This is the remote node address of the remote bridge. You do not need to specify its address.
6. This is the Remote Link ID.
7. This is the Target Node Address.

```
Type:                                Peer–to–Peer
Read/Write:                                Read
Target Device:                          500 CPU
Local/Remote:                            Remote
Control Block:                     user specified
Channel:                                      1
Target Node (decimal):                        3
Remote Bridge Link Id:                        2
Remote Bridge Node Address <dec>:             8
Local Bridge Node Address <dec>:              4
Destination/Source File Addr:      user specified
Target Src/Dst File Address:       user specified
Message Length In Elements:        user specified
Message Timeout (seconds):         user specified
```

Example 4 – Passthru via DF1 Full-Duplex Channel 0 of the SLC 5/04 Processor



SLC 5/04 Processor (A) to SLC 5/04 Processor (D) via two SLC 5/04 Processors (Passthru using Channel 0 DF1 Full-Duplex)

```
Type:                                Peer-to-Peer
Read/Write:                             Read
Target Device:                       500 CPU
Local/Remote:                         Remote
Control Block:                    user specified
Channel:                                 1
Target Node (decimal):                  63
Remote Bridge Link Id:                   1
Remote Bridge Node Address <dec>:        0
Local Bridge Node Address <dec>:         2
Destination/Source File Addr:     user specified
Target Src/Dst File Address:      user specified
Message Length In Elements:       user specified
Message Timeout (seconds):        user specified
```

Note *Improper configuration may cause data to be written to or read from an unintended processor. Make sure that all parameters and channel link IDs are set correctly, as well as bit S:34/5 being set in* ***both*** *passthru processors..*

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/04 processor on the DH+ network.

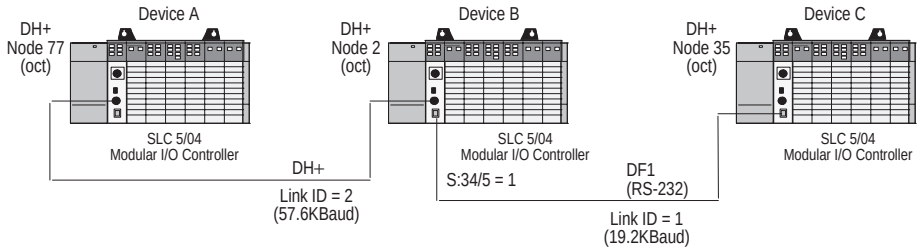
Target Node is the SLC 5/04 processor at node address 77 (63 decimal).

Remote Bridge Link ID is the link ID of the DF1 full-duplex RS-232 connection between the passthru SLC 5/04 processors' channel 0 (Link ID 1).

Remote Bridge Node Address is set to 0 (not used) because Channel 0 is DF1 full-duplex.

Local Bridge Node Address is set to 2 since this is the DH+ node address used by the local SLC 5/04 passthru processor.

Example 5 – Passthru via DH+ Channel 0 of the SLC 5/04 Processor



Note *Device B has S:34/5 set to 1. Device C must have S:34/5 cleared. Otherwise, Device C would attempt to passthru this message packet to node 0 on its Channel 1 DH+ network instead of responding to the message.*

SLC 5/04 Processor (A) to SLC 5/04 Processor (C) via a single SLC 5/04 Processor (Passthru using Channel 0 DF1)

```
Type:                              Peer-to-Peer
Read/Write:                              Read
Target Device:                        500 CPU
Local/Remote:                          Remote
Control Block:                  user specified
Channel:                                   1
Target Node:                               0
Remote Bridge Link Id <dec>:               1
Remote Bridge Node Address:                0
Local Bridge Node Address:                 2
Destination/Source File Addr:   user specified
Target Src/Dst File Address:    user specified
Message Length In Elements:     user specified
Message Timeout (seconds):      user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/04 processor on the DH+ network.

Target Node is the SLC 5/04 processor at node address 0 (with DF1 full-duplex, any valid node address would work here).

Remote Bridge Link ID is the link ID of the passthru SLC 5/04 processor's channel 0 (Link ID 1).

Remote Bridge Node Address is set to 0 (not used) because Channel 0 is DF1 full-duplex.

Local Bridge Node Address is set to 2 since this is the DH+ node address used by the local SLC 5/04 passthru processor.

SLC 5/04 Processor (C) to SLC 5/04 Processor (A) via a single SLC 5/04 Processor (Passthru using Channel 0 DF1)

```
Type:                              Peer–to–Peer
Read/Write:                              Read
Target Device:                       500 CPU
Local/Remote:                          Local
Control Block:                   user specified
Channel:                                    0

Target Node (decimal):                     63
Destination/Source File Addr:    user specified
Target Src/Dst File Address:     user specified
Message Length In Elements:      user specified
Message Timeout (seconds):       user specified
```

Comments

Channel is set to 0 since the originating command is initiated by an SLC 5/04 processor, connected via DF1 full-duplex.

Target Node is the SLC 5/04 processor at node address 63 decimal (77 octal).

SLC 5/04 Processor (C) to SLC 5/04 Processor (B) when Channel 0 DF1 Passthru is Enabled

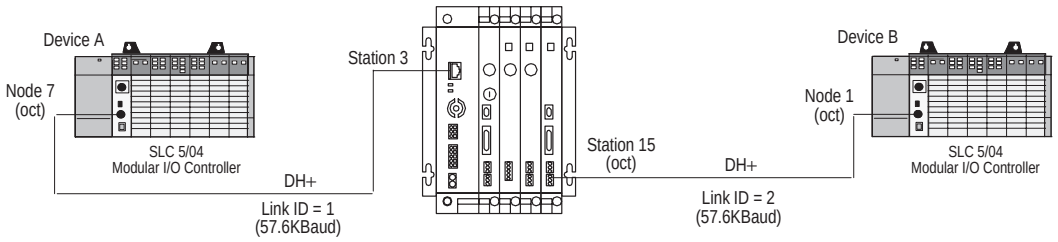
```
Type:                              Peer–to–Peer
Read/Write:                              Read
Target Device:                       500 CPU
Local/Remote:                          Local
Control Block:                   user specified
Channel:                                    0
Target Node (decimal):                      2


Destination/Source File Addr:    user specified
Target Src/Dst File Address:     user specified
Message Length In Elements:      user specified
Message Timeout (seconds):       user specified
```

Comments

Channel is set to 0 since the originating command is initiated by an SLC 5/04 processor, connected via DF1 full-duplex.

Target Node is the SLC 5/04 processor at DH+ node address 2.

Example 6 – Passthu using a Pyramid Integrator for Routing a message instruction



SLC 5/04 Processor (B) to SLC 5/04 Processor (A) via Pyramid Integrator using PI routing

```
Type:                                Peer–to–Peer
Read/Write:                                Read
Target Device:                          500 CPU
Local/Remote:                            Remote
Control Block:                     user specified
Channel:                                      1
Target Node:                                  7
Remote Bridge Link Id <dec>:                  1
Remote Bridge Node Address:                   0
Local Bridge Node Address:                   13
Destination/Source File Addr:      user specified
Target Src/Dst File Address:       user specified
Message Length In Elements:        user specified
Message Timeout (seconds):         user specified
```

Comments

Channel is set to 1 since the originating command is initiated by an SLC 5/04 processor on the DH+ network.

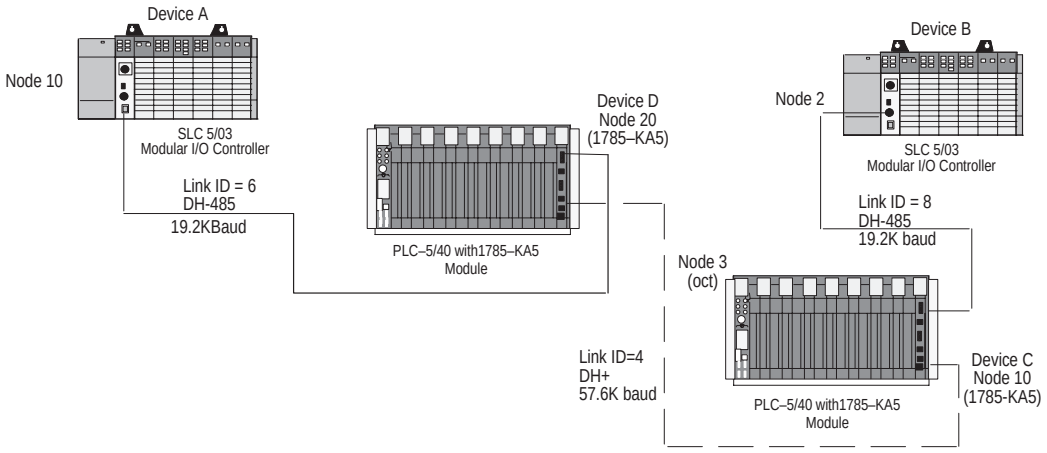
Target Node is the SLC 5/04 processor at node address 7.

Remote Bridge Link ID is the link ID of the remote DH+ network with the SLC 5/04 processor (Channel 1, Link ID 1) and the Pyramid Integrator.

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 13 decimal (15 octal) since this is the DH+ node address of the Pyramid Integrator on the local DH+ network.

Example 7 – Passthu using Two 1785-KA5s



SLC 5/03 Processor (A) to an SLC 5/03 Processor (B) (Using two 1785–KA5s)

```
Type:                              Peer–to–Peer
Read/Write:                       Read or Write
Target Device:                    485CIF or 500 CPU
Local/Remote:                            Remote
Control Block:                    user specified
Channel:                                     1
Target Node:                                 2
Remote Bridge Link Id:                       8
Remote Bridge Node Address:                  0
Local Bridge Node Address:                  20
Destination/Source File Addr:     user specified
Target Src/Dst File Address:      user specified
Message Length In Elements:       user specified
Message Timeout (seconds):        user specified
```

Comments

Channel is set to 1 since the command is sent from the SLC 5/03's DH-485 channel onto local Link ID 4.

Target Node is set to 2 since this is the DH-485 address the destination device resides at on the destination link (Link ID 8).

Remote Bridge Link ID is set to 8 since this is the destination link that the destination device resides on.

Remote Bridge Node Address is set to 0 (not used) because communication is from one Internet-capable device to another Internet-capable device.

Local Bridge Node Address is set to 20 since it is the bridge device (Link ID 4) that the command is to be sent through (device D).

Service Communications (SVC)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
	✓	✓	✓	✓	✓

Using an SLC 5/02 Processor

—(SVC)—

Output Instruction

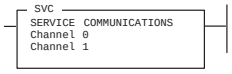
The SVC instruction is an output instruction that has no programming parameters. When it is evaluated as true, the program scan is interrupted to execute the service communications part of the operating cycle. The scan then resumes at the instruction following the SVC instruction. Use this instruction to enhance the communication performance of your SLC 5/02 processor.

You are not allowed to place an SVC instruction in a STI interrupt, I/O interrupt, or user fault subroutine.

The following status bits allow you to customize or monitor communications servicing. Refer to chapter 1 in this manual for additional information on the status file.

- S:2/5 DH-485 Incoming Command Pending
- S:2/6 DH-485 Message Reply Pending
- S:2/7 DH-485 Outgoing Message Command Pending
- S:2/15 DH-485 Communications Servicing Selection

Using SLC 5/03 and Higher Processors



Output Instruction

When using SLC 5/03 and higher processors, the SVC instruction operates as described above. These processors also allow you to select a specific communication channel (0, 1, or both) to be serviced. You are not allowed to place an SVC instruction in a Fault, DII, STI, or I/O Event subroutine.

- SLC 5/03 processor
  - channel 0 is RS-232/DF1 Full-Duplex or Half-Duplex (master or slave), DH-485, or ASCII
  - channel 1 is DH-485
- SLC 5/04 processor
  - channel 0 is RS-232/DF1 Full-Duplex or Half-Duplex (master or slave), DH-485, or ASCII
  - channel 1 is DH+
- SLC 5/05 processor
  - channel 0 is RS-232/DF1 Full-Duplex or Half-Duplex (master or slave), DH-485, or ASCII
  - channel 1 is Ethernet

The following status bits allow you to customize or monitor communications servicing. Refer to appendix B in this manual for additional information about the status file.

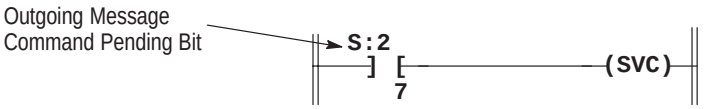
Channel 1		Channel 0	
S:2/5	Incoming Command Pending Bit	S:33/0	Incoming Command Pending
S:2/6	Message Reply Pending Bit	S:33/1	Message Reply Pending
S:2/7	Outgoing Message Command Pending Bit	S:33/2	Outgoing Message Command Pending
S:2/15	Communications Servicing Selection Bit	S:33/5	Communications Servicing Selection
S:33/7	Message Servicing Selection Bit	S:33/6	Message Servicing Selection

Channel Servicing

When a channel is not selected to be serviced by the SVC instruction, that channel is serviced normally at the end of the scan.

Application Example

The SVC instruction is used when you want to execute a communication function, such as transmitting a message, prior to the normal service communication portion of the operating scan. The following example shows how to selectively use the SVC instruction.



You can place this rung after a message write instruction. S:2/7 is set when the message instruction is enabled and waiting (provided no message is currently being transmitted). When S:2/7 is set, the SVC instruction is evaluated as true and the program scan is interrupted to execute the service communications portion of the operating scan. The scan then resumes at the instruction following the SVC instruction.

This simple example assumes that the Comms Servicing Selection bit S:2/15 is clear and that this is the only active MSG instruction.

Note *You may program the SVC instruction unconditionally across the rungs. This is the normal programming technique for the SVC instruction.*

9 MicroLogix Communication Instruction

This chapter contains information about communications and the message (MSG) instruction. Specifically, this chapter contains information on:

- types of communication
- what the MSG instruction symbol looks like
- typical execution time for the MSG instruction
- how to use the MSG instruction
- application examples and timing diagrams

Note *Only Series C or later MicroLogix 1000 controllers and Series A or later MicroLogix 1000 analog controllers support the MSG instruction.*

Message Instruction

Instruction		Purpose	Page
Mnemonic	Name		
MSG	Message Read/Write	This instruction transfers data from one node to another via the communication port. When the instruction is enabled, the message is sent to a communication buffer. Replies are processed at the end of scan.	9–2

Types of Communication

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓						

Communication is the ability of a device to send data or status to other devices. This capability typically falls into one of two categories: master/sender or slave/receiver. Each of these are described below:

Initiator (Master) Communication

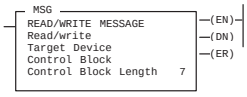
Initiator products can initiate communication, which includes requesting information from other devices (reading) or sending information to other products (writing). In addition, initiator products are usually capable of replying to other devices when they make requests to read information. The Series C or later MicroLogix 1000 controllers and the Series A or later MicroLogix 1000 analog controllers are in this class.

Initiator products can initiate communication with other intitiator products (peer-to-peer communication) or with responder products (initiator-to-responder communication).

Responder (Slave) Communication

Responder products can only reply to other products. These devices are not capable of initiating an exchange of data; they only reply to requests made from initiator products. The Series A and B MicroLogix 1000 controllers are in this class.

Message Instruction (MSG)



Execution Times (µsec) when:	
True	False
180①	48

The MSG is an output instruction that allows the controller to initiate an exchange of data with other devices. The relationship with the other devices can be either peer-to-peer communication or master-to-slave communication. The type of communication required by a particular application determines the programming configuration requirements of the MSG instruction.

① This only includes the amount of time needed to set up the operation requested. It does not include the time it takes to service the actual communication, as this time varies with each network configuration. As an example, 144ms is the actual communication service time for the following configuration: 3 nodes on DH-485 (2=MicroLogix 1000 programmable controllers and 1=PLC-500 A.I. Series™ programming software), running at 19.2K baud, with 2 words per transfer.

Entering Parameters

After you place the MSG instruction on a rung, specify whether the message is to be a read or write. Then specify the target device and the control block for the MSG instruction.

- **Read/Write** – read indicates that the local processor (processor in which the instruction is located) is receiving data; write indicates that the processor is sending data.
- **Target Device** – identifies the type of command used to establish communication. The target device can be a MicroLogix 1000 controller or SLC family processor using SLC commands, or a common interface file by selecting the CIF format. Valid options are:
  - SLC500/ML1000 – Allows communication between a MicroLogix 1000 controller and any other MicroLogix 1000 controller or SLC 500 family processor.
  - 485CIF – (common interface file) Allows communication between a MicroLogix 1000 controller and a non-MicroLogix 1000/SLC 500 device. The CIF data is automatically delivered to integer file 9 in SLC 500 processors or file 7 in MicroLogix 1000 controllers. The 485CIF protocol is also used for PLC-2 type messages.
- **Control Block Address** – an integer file address that you select. It consists of 7 integer words, containing the status bits, target file address, and other data associated with the MSG instruction.
- **Control Block Length** – fixed at seven elements. This field cannot be altered.

Note

When running a MicroLogix 1000 program on an SLC 5/03 or SLC 5/04 processor, the MSG control block length increases from 7 to 14 words. If you plan to run a MicroLogix 1000 program with one of these processors, make sure that the program has at least 7 unused words following each MSG control block.

The table that follows illustrates combinations of message types and target devices and their valid file types.

Command Type	Message Type	Initiating Device	Valid File Types	Target Device①②③	Valid File Types
SLC500/ML1000	Write	MicroLogix 1000	O,I,S,B,T,C,R,N	MicroLogix 1000	O,I,S,B,T,C,R,N
SLC500/ML1000	Read	MicroLogix 1000	O,I,S,B,T,C,R,N	MicroLogix 1000	O,I,S,B,T,C,R,N
CIF	Write	MicroLogix 1000	O,I,S,B,T,C,R,N	MicroLogix 1000	N7
CIF	Read	MicroLogix 1000	O,I,S,B,T,C,R,N	MicroLogix 1000	N7
SLC500/ML1000	Write	MicroLogix 1000	O,I,S,B,T,C,R,N	SLC 500	O④,I④,S,B,T,C,R,N
SLC500/ML1000	Read	MicroLogix 1000	O,I,S,B,T,C,R,N	SLC 500	O④,I④,S,B,T,C,R,N
CIF	Write	MicroLogix 1000	O,I,S,B,T,C,R,N	SLC 500	N9
CIF	Read	MicroLogix 1000	O,I,S,B,T,C,R,N	SLC 500	N9

① The DF1 Full-Duplex protocol can be used if the target device supports it. Such devices include MicroLogix 1000 controllers (any series), SLC 5/03 and SLC 5/04 processors, and PLC-5 processors (CIF command type only).

② The DH-485 protocol can be used if the target device supports it. Such devices include MicroLogix 1000 controllers (except for Series A and B controllers) and SLC 500, SLC 5/01, SLC 5/02, SLC 5/03, or SLC 5/04 processors.

③ The DF1 Half-Duplex protocol can also be used with Series D and analog MicroLogix 1000 controllers, as well as SLC 5/03 and SLC 5/04 processors, but a master is required.

④ SLC 500, SLC 5/01, and SLC 5/02 processors do not support O or I file access from a MSG instruction. SLC 5/03 and SLC 5/04 processors do support O and I file access, but only when unprotected.

Control Block Layout

The control block layouts shown below illustrate SLC500/ML1000 type messages.

Control Block Layout – SLC500/ML1000

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	Word
EN ST DN ER EW NR TO | Error Code	0
| Node Number	1
Reserved for length (in elements)	2
File Number	3
File Type (O, I, S, B, T, C, R, N)	4
Element Number	5
Subelement Number	6

Control Block Layout – 485CIF

15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00	Word
EN ST DN ER EW NR TO | Error Code	0
| Node Number	1
Reserved for Length (in elements)	2
Offset Bytes	3
Not used	4
Not used	5
Not used	6

Using Status Bits

```
Read/Write:                          READ          ignore if timed out:   0  TO
 Target Device:            SLC500/ML1000                to be retried:   0  NR
 Control Block:                    N7:0           awaiting execution:   0  EW
 Local Destination File Address:    ***
 Target Node:                         0                        error:   0  ER
 Target File Address:               ***                message done:   0  DN
 Message Length in elements         ***         message transmitting:   0  ST
                                                     message enabled:   0  EN

                                                 control bit address:    N7:0/8

 ERROR CODE:  0
 Error Code Desc:
```

MSG Instruction Status Bits

The right column in the display above lists the various MSG instruction status bits. These are explained below:

- **Time Out Bit TO (bit 08)** Temporarily set this bit (1) to clear an existing MSG instruction. This bit has no effect unless the ST bit has first been set due to receiving an ACK (acknowlege). Your application must supply its own timeout value. This bit is reset on any false-to-true rung transition.
- **Negative Response Bit NR (bit 09)** is set if the target processor is responding to your message, but can not process the message at the present time. The NR bit is reset at the next false-to-true rung transition that has a transmit buffer available. It is used to determine when to send retries. The ER bit is also set at this time. Use this feedback to initiate a retry of your message at a later time.
- **Enabled and Waiting Bit EW (bit 10)** is set on any false-to-true transition. This bit is reset when an ACK or NAK (no acknowledge) is received, or on any false rung instruction execution.

Note *The operation of the EW bit has changed since Series C.*

- **Error Bit ER (bit 12)** is set when message transmission has failed. The ER bit is reset the next time the rung goes from false to true.
- **Done Bit DN (bit 13)** is set when the message is transmitted successfully. The DN bit is reset (cleared) the next time the rung goes from false to true.

- **Start Bit ST (bit 14)** is set when the processor receives acknowledgement from the target device. This identifies that the target device has started to process the MSG request. The ST bit is reset when the DN, ER, or TO bit is set or on a false-to-true rung transition.
- **Enable Bit EN (bit 15)** is set only if the transmit buffer is available. If the transmit buffer is not available, the EN flag remains false. When the transmit buffer becomes available, the EN flag goes true. It remains set until the next false rung execution after the MSG completes (DN bit set) or an error occurs (ER bit set).

Note *The operation of the EN bit has changed with Series C controllers.*

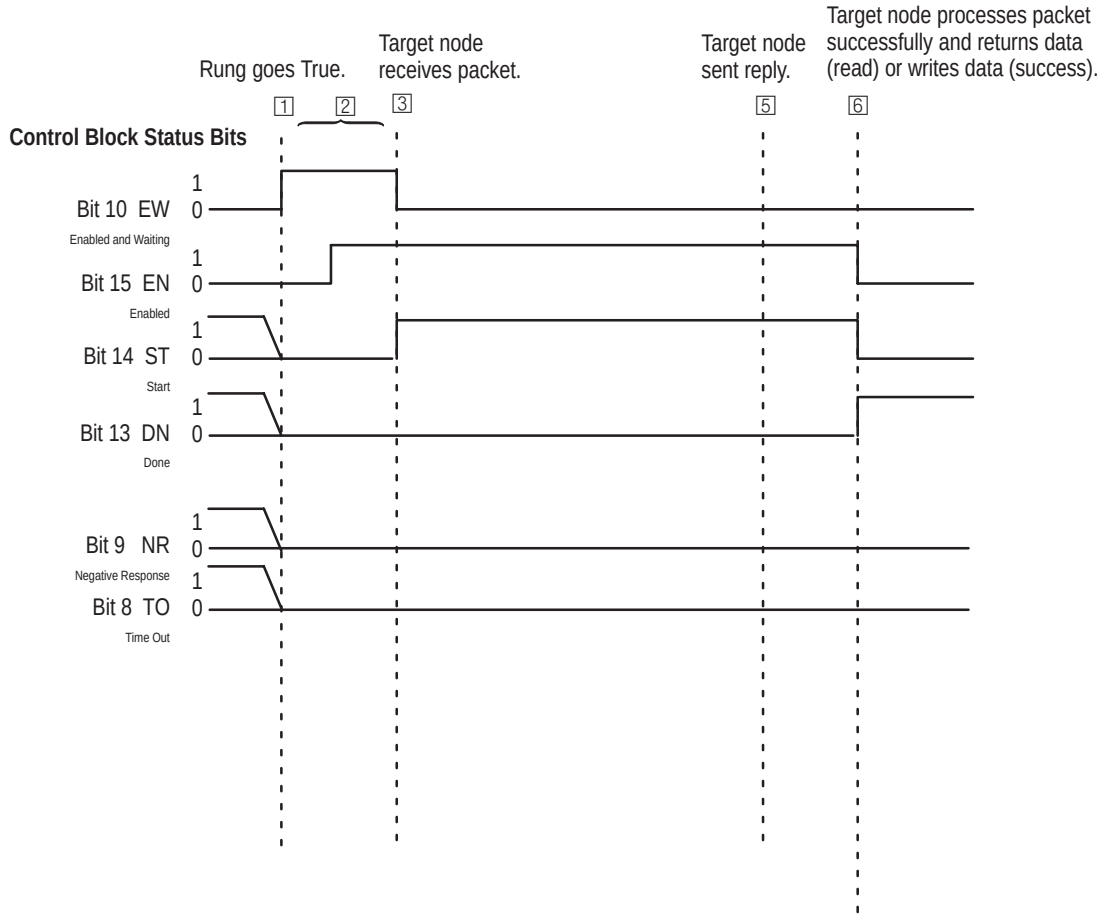
Controller Communication Status Bit

When using the MSG instruction, you should also use the following controller communication status bit:

Active Protocol Bit (S:0/11) – This is a read-only bit that indicates which communication protocol is currently enabled or functioning; where 0 = DF1 (default) and 1 = DH-485. Use this bit in your program to restrict message operation to the specific protocol in use.

Timing Diagram for a Successful MSG Instruction

The following section illustrates a successful timing diagram for a MicroLogix 1000 Series D or later, or analog Series A, MSG instruction.



1 The **EW** bit is set (1) and the **ST, DN, NR**, and **TO** flags are cleared. If the transmit buffer is not available, the EN flag remains false (0).

2 When rung conditions go true and the transmit buffer becomes available, the **EN** flag goes true (1). The EN bit remains set until either the **DN**, **ER**, or **TO** bit is set. The **TO** bit has no effect unless the ST bit has first been set.

3 If the Target Node successfully receives the MSG packet, it sends back an ACK (an acknowledge). The ACK causes the processor to clear bit S:2/7. Note that the Target Node has not yet examined the MSG packet to see if it understands your request. It is replying to the initial connection.

At the next end of scan, the **EW** bit is cleared (0) and the **ST** bit is set (1). Once the **ST** bit is set, the processor will wait indefinitely for a reply from the Target Node. The Target Node is not required to respond within any given time frame. At this time, no other MSG instruction will be serviced.

Note

*If the Target Node faults or power cycles during the time frame of a MSG transaction, you will never receive a reply. This is why it is recommended you use a timer in conjunction with the **TO** bit to clear any pending instructions. (When the **TO** bit is set [1] it clears pending messages.) Typically message transactions are completed within a couple of seconds. It is up to the programmer to determine how long to wait before clearing the buffer and then re-transmitting.*

Step 4 is not shown in the timing diagram.

4 If you do not receive an ACK, step 3 does not occur. Instead a NAK (no acknowledge) is received. When this happens, the **ST** bit remains clear. A NAK indicates:

- either the Target Node is not there,
- it does not respond,
- it is too busy, or
- it received a corrupt MSG packet.

When a NAK occurs, the **EW** bit is cleared. (Note that the NR bit will only be set for DH-485 and NAK conditions. An crror code 02H, Target Node is busy, is received which causes the **NR** bit to be set.) The **ER** bit is also set which indicates that the MSG instruction failed.

Monitor the **NR** bit. If it is set, indicating that the Target Node is busy, you may want to initiate some other process (e.g., an alarm or a retry later). The **NR** bit is cleared when the rung logic preceding the MSG changes from false to true.

5 When an ACK occurs, the Target Node sends one of three responses shown in Step 6.

6 Following the successful receipt of the packet, the Target Node sends a reply packet. The reply packet will contain one of the following responses:

- I have successfully performed your write request.
- I have successfully performed your read request, and here is your data.
- I have not performed your request because of an error.

At the next end of scan, following the Target Node's reply, the MicroLogix 1000 controller examines the MSG packet from the target device. If the reply contains "I have successfully performed your write request," the **DN** bit is set and the **ST** bit is cleared. The MSG instruction is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

If the reply contains "I have successfully performed your read request, and here is your data," the data is written to the appropriate data table, the **DN** bit is set, and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

If the reply contains "I have not performed your request, because of an error," the **ER** bit is set and the **ST** bit is cleared. The MSG instruction function is complete. If the MSG rung is false, the **EN** bit is cleared the next time the MSG instruction is scanned.

MSG Instruction Error Codes

Note *Any MSG instruction that is in progress during a network protocol switch will not be processed and will be discarded.*

When an error condition occurs, the error code is stored in the *lower byte* of the first control word assigned to the MSG instruction.

Error Code	Description of Error Condition
02H	Target node is busy.
03H	Target node cannot respond because message is too large.
04H	Target node cannot respond because it does not understand the command parameters OR the control block may have been inadvertently modified.
05H	Local processor is offline (possible duplicate node situation).
06H	Target node cannot respond because requested function is not available.
07H	Target node does not respond.
08H	Target node cannot respond.
09H	Local modem connection has been lost.
0AH	Buffer unavailable to receive SRD reply.
0BH	Target node does not accept this type of MSG instruction.
0CH	Received a master link reset.
10H	Target node cannot respond because of incorrect command parameters or unsupported command.
15H	Local channel configuration parameter error exists.
18H	Broadcast (Node Address 255) is not supported.
1AH①	Target node cannot respond because another node is file owner (has sole file access).
1BH①	Target node cannot respond because another node is program owner (has sole access to all files).
37H	Message timed out in local processor.
39H	Message was discarded due to a communication protocol switch.
3AH	Reply from target is invalid.
50H	Target node is out of memory.
60H	Target node cannot respond because file is protected.
E7H	Target node cannot respond because length requested is too large.
EBH	Target node cannot respond because target node denies access.
ECH	Target node cannot respond because requested function is currently unavailable.
FAH	Target node cannot respond because another node is file owner (has sole file access).
FBH	Target node cannot respond because another node is program owner (has sole access to all files).

① Error codes 1A and 1B valid for Series C only.

Note

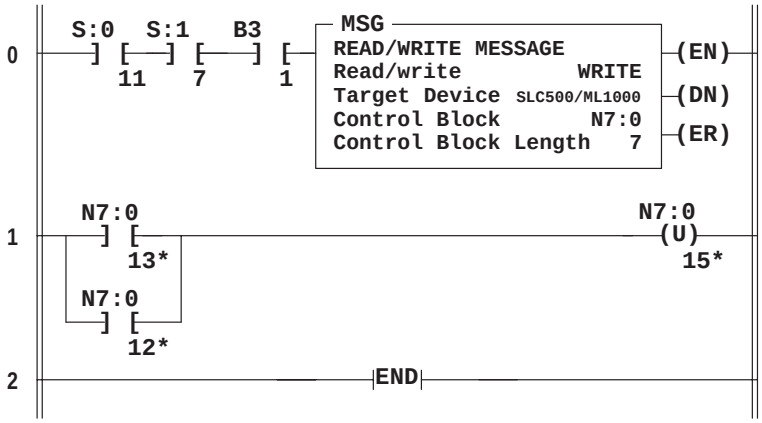
For 1770–6.5.16 DF1 Protocol and Command Set users:

The MSG error code reflects the STS field of the reply to your MSG instruction.
Codes E0 – EF represent EXT STS codes 0 – F.
Codes F0 – FC represent EXT STS codes 10 – 1C.

Application Examples that Use the MSG Instruction

Example 1

Application example 1 shows how you can implement continuous operation of a message instruction.



* MSG instruction
status bits:
12 = ER
13 = DN
15 = EN

Operation Notes

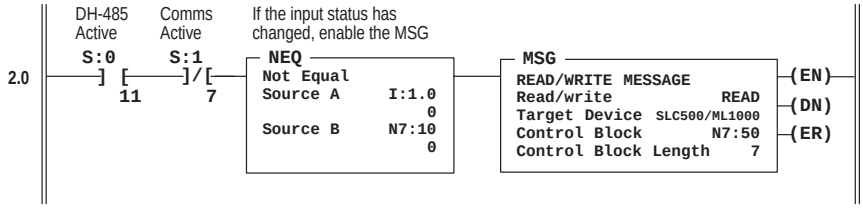
Bit S:0/11 ensures that the MSG instruction is only processed when the active protocol is DH-485. Bit S:1/7 ensures that DH-485 is communicating before sending the MSG. Bit B3/1 enables the MSG instruction. When the MSG instruction done bit (N7:0/13) is set, it unlatches the MSG enable bit (N7:0/15) so that the MSG instruction is re-enabled in the next scan. This provides continuous operation.

The MSG error bit also unlatches the enable bit. This provides continuous operation even if an error occurs.

Example 2

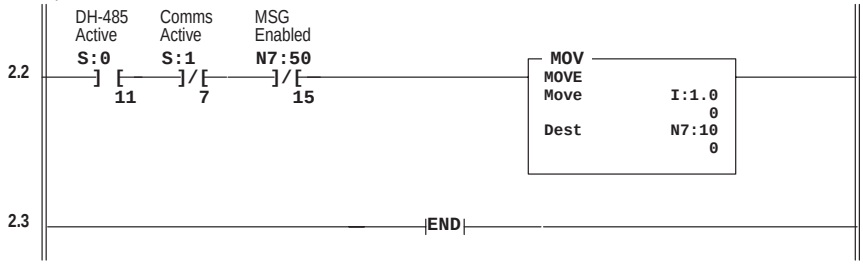
Application example 2 involves a MicroLogix 1000 controller transmitting its first input word to another MicroLogix 1000 controller. This is commonly referred to as "change of state" or "report on exception" messaging. Using this type of logic significantly reduces network traffic, which in turn significantly improves network throughput.

This is the message control rung. The logic preceding the MSG instruction on this rung dictates when the MSG instruction is processed. In this example, the MSG instruction is only processed when the active protocol is DH-485 and when there is no other communication. Once the MSG instruction is enabled, it locks itself into operation regardless of the preceding logic on the rung.



This rung controls when the MSG instruction is unlatched or reset. The MSG instruction must be reset before it can re-transmit new information. Either of the following two conditions resets the MSG instruction: 1) when communication to the target device have been completed successfully, or 2) when an error is detected in the communication sequence (occurs after all retries have been exhausted). Using the error bit to reset the MSG is primarily used to stop the MSG instruction from being totally locked out.

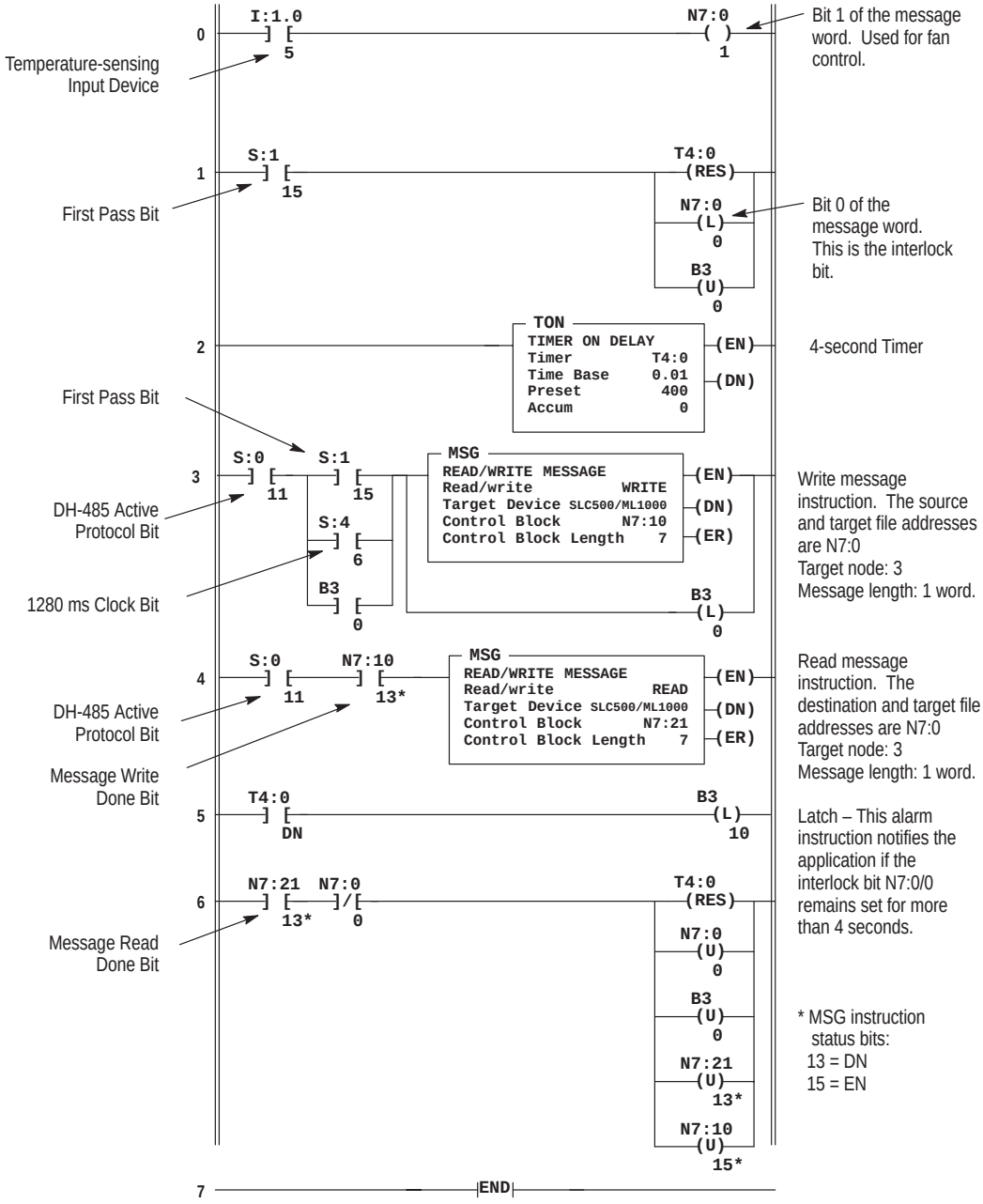
This rung is used to setup the "report by exception" operation. This move command updates N7:10, by making it identical to I:1.0. When the processor starts a new scan sequence (when rung 2.0 is scanned,) it updates (reads) the input image. If an input has changed from the previous scan, the NEQ instruction is true and MSG is processed. The MSG Enabled bit ensures that the MOV is not processed until after the MSG is successfully completed. This minimizes the chances that input changes are missed during MSG operation.



Example 3

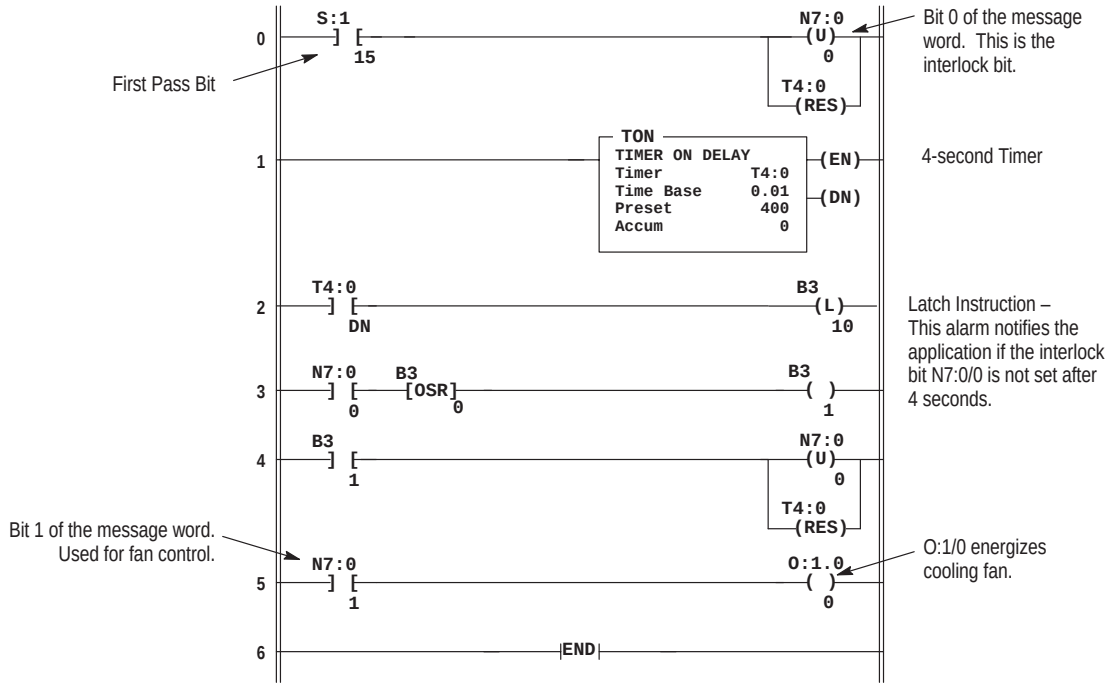
Application example 3 involves a MicroLogix 1000 controller and an SLC 5/01 processor communicating on a DH-485 network. Interlocking is provided to verify data transfer and to shut down both processors if communication fails.

A temperature-sensing device, connected as an input to the MicroLogix 1000 controller, controls the on-off operation of a cooling fan, connected as an output to the SLC 5/01 processor. The MicroLogix 1000 and SLC 5/01 ladder programs are explained on the following pages.



Operation notes appear on the following page.

Program File 2 of SLC 5/01 Processor at Node 3



Operation Notes, MicroLogix 1000 and SLC 5/01 programs

Message instruction parameters: N7:0 is the message word. It is the target file address (SLC 5/01 processor) and the local source and destination addresses (MicroLogix 1000 controller) in the message instructions.

N7:0/0 of the message word is the interlock bit; it is written to the 5/01 processor as a 1 (set) and read from the SLC 5/01 processor as a 0 (reset).

N7:0/1 of the message word controls cooling fan operation; it is written to the SLC 5/01 processor as a 1 (set) if cooling is required or as a 0 (reset) if cooling is not required. It is read from the SLC 5/01 processor as either 1 or 0.

Word N7:0 should have a value of 1 or 3 during the message write execution. N7:0 should have a value of 0 or 2 during the message read execution.

Program initialization: The first pass bit S:1/15 initializes the ladder programs on run mode entry.

MicroLogix 1000 controller: N7:0/0 is latched; timer T4:0 is reset; B3/0 is unlatched (rung 1), then latched (rung 3).
SLC 5/01 processor: N7:0/0 is unlatched; timer T4:0 is reset.

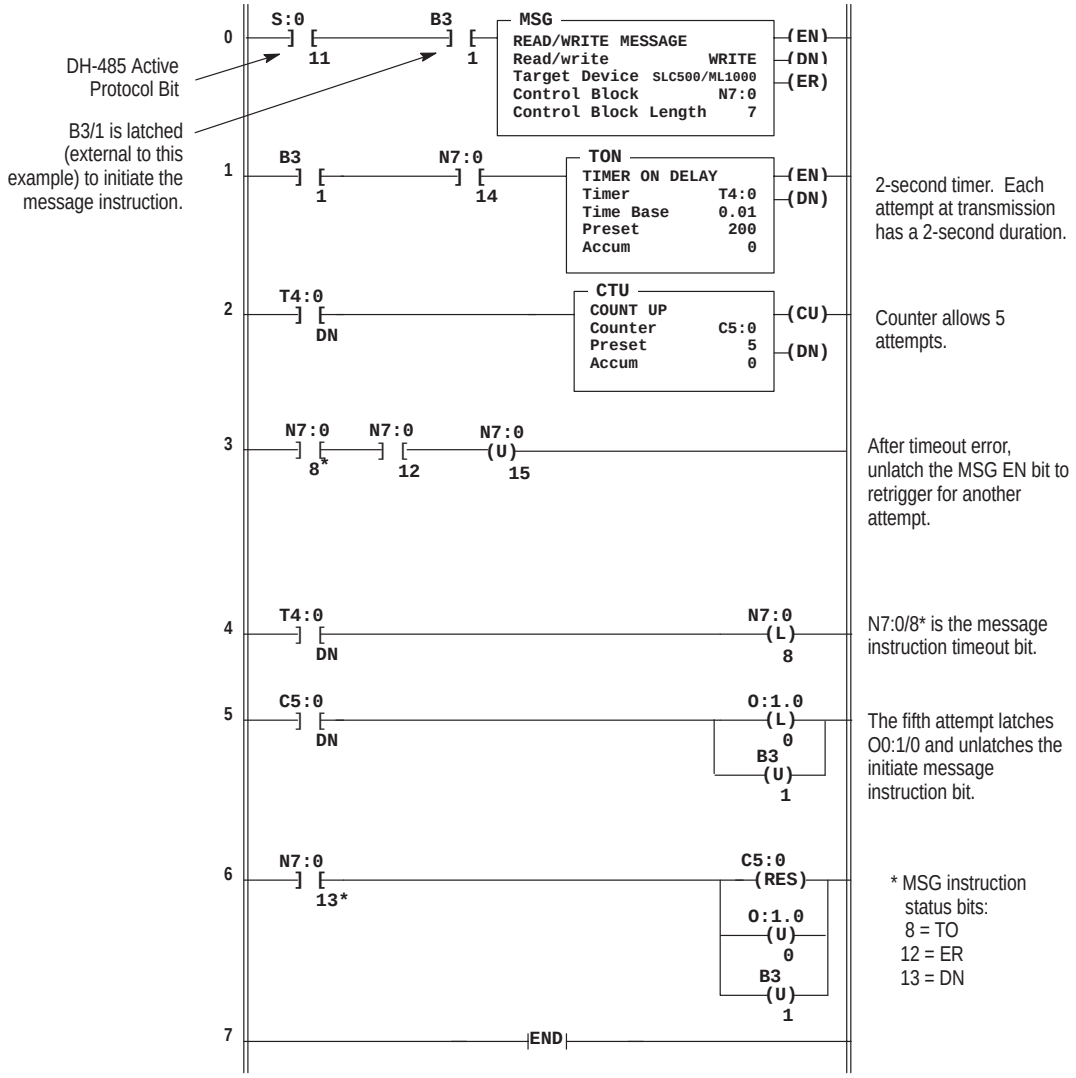
Message instruction operation: The message write instruction in the MicroLogix 1000 controller is initiated every 1280 ms by clock bit S:4/6. The done bit of the message write instruction initiates the message read instruction.

B3/0 latches the message write instruction. B3/0 is unlatched when the message read instruction done bit is set, provided that the interlock bit N7:0/0 is reset.

Communication failure: In the MicroLogix 1000 controller, bit B3/10 becomes set if interlock bit N7:0/0 remains set (1) for more than 4 seconds. In the SLC 5/01 processor, bit B3/10 becomes set if interlock bit N7:0/0 remains set (1) for more than 4 seconds. Your application can detect this event, take appropriate action, then unlatch bit B3/10.

Example 4

Application example 4 shows you how to use the timeout bit to disable an active message instruction. In this example, an output is energized after five unsuccessful attempts (two seconds duration) to transmit a message.



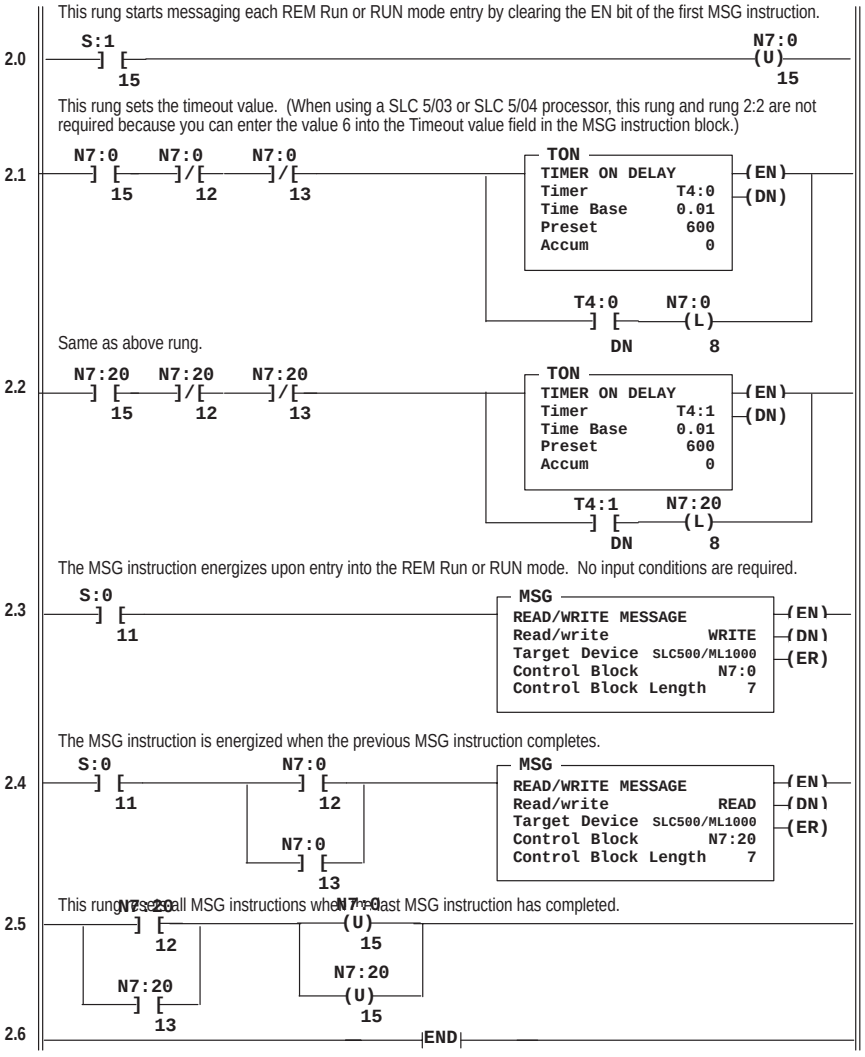
Operation Notes

The timeout bit is latched (rung 4) after a period of 2 seconds. This clears the message instruction from processor control on the next scan. The message instruction is then re-enabled for a second attempt at transmission. After 5 attempts, O:1/0 is latched and B3/1 is unlatched.

A successful attempt at transmission resets the counter, unlatches O:1/0, and unlatches B3/1.

Example 5

Application example 5 shows you how to link message instructions together to transmit serially, one after another. In this example, a MSG Write is followed by a MSG Read, which causes the serial transmission.

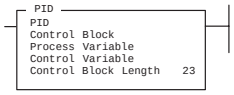


10 Proportional Integral Derivative Instruction

This chapter describes the Proportional Integral Derivative (PID) instruction.

Overview

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓	✓



Output Instruction

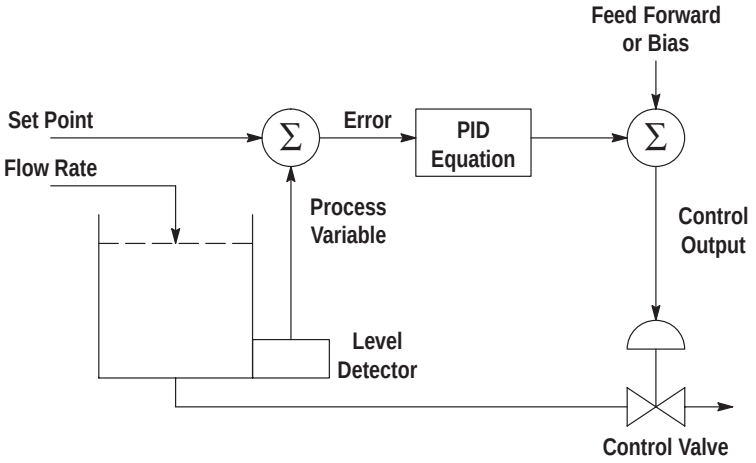
This is an output instruction that controls physical properties such as temperature, pressure, liquid level, or flow rate using process loops.

The PID instruction normally controls a closed loop using inputs from an analog input module and providing an output to an analog output module. For temperature control, you can convert the analog output to a time proportioning on/off output for driving a heater or cooling unit. An example appears on pages 10–14 through 10–16.

The PID instruction can be operated in the timed mode or the STI mode. In the timed mode, the instruction updates its output periodically at a user-selectable rate. In the STI mode, the instruction should be placed in an STI interrupt subroutine. It then updates its output every time the STI subroutine is scanned. The STI time interval and the PID loop update rate must be the same in order for the equation to execute properly.

The PID Concept

PID closed loop control holds a process variable at a desired set point. A flow rate/fluid level example is shown below.



The PID equation controls the process by sending an output signal to the control valve. The greater the error between the setpoint and process variable input, the greater the output signal, and vice versa. An additional value (feed forward or bias) can be added to the control output as an offset. The result of PID calculation (control variable) drives the process variable you are controlling toward the set point.

The PID Equation

The PID instruction uses the following algorithm:

Standard equation with dependent gains:

$$Output = K_C\ [(E) + 1/T_I \int (E)dt + T_D \cdot D(PV)/dt] + bias$$

Standard Gains constants are:

Term	Range (Low to High)	Reference
Controller Gain K_C	0.1 to 25.5 (dimensionless) 0.01 to 327.67 (dimensionless)①	Proportional
Reset Term $1/T_I$	25.5 to 0.1 (minutes per repeat) 327.67 to 0.01 (minutes per repeat)①	Integral
Rate Term T_D	0.01 to 2.55 (minutes) 0.01 to 327.67 (minutes)①	Derivative

① Applies to SLC 5/03 and higher processors PID ranges when bit Reset and Gain Range (RG) bit is set to 1.

The derivative term (rate) provides smoothing by means of a low-pass filter. The cutoff frequency of the filter is 16 times greater than the corner frequency of the derivative term.

Entering Parameters

Normally, you place the PID instruction on a rung without conditional logic. The output remains at its last value when the rung is false. The integral term is also cleared when the rung is false.

Note *The PID instruction is an integer-only type of PID algorithm and does not allow you to enter floating point values for any of its parameters. So, if you attempt to move a floating point value to one of the PID parameters using ladder logic, a floating point-to-integer conversion occurs.*

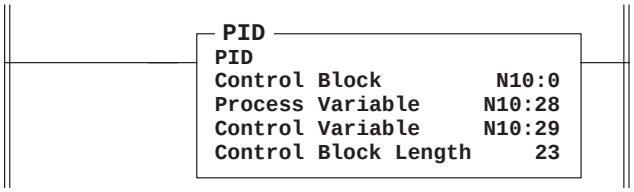
During programming, you enter the Control Block, Process Variable, and Control Variable addresses after you have placed the PID instruction on a rung:

- **Control Block** is a file that stores the data required to operate the instruction. The file length is fixed at 23 words and should be entered as an integer file address. For example, an entry of N10:0 will allocate elements N10:0 through N10:22. The control block layout is shown on page 10–10.

  Do not write to control block addresses with other instructions in your program except as described later in this chapter. If you are re-using a block of data which was previously allocated for some other use, it is good practice to first zero the data. We recommend that you use a unique data file to contain your PID control blocks (for example, N10:0). This avoids accidental re-use of the PID control block addresses by other instructions in your program.

- **Process Variable PV** is an element address that stores the process input value. This address can be the location of the analog input word where the value of the input A/D is stored. This value could also be an integer value if you choose to pre-scale your input value to the range 0–16383.
- **Control Variable CV** is an element address that stores the output of the PID instruction. The output value ranges from 0 to 16383, with 16383 being the 100% "on" value. This is normally an integer value, so that you can scale the PID output range to the particular analog range your application requires.

The figure below shows a PID instruction with typical addresses for these parameters entered:




```
          auto/manual: MANUAL*                                  time mode Bit:  1 TM
                 mode: TIMED *                                auto/manual bit:  1 AM
              control: E=SP-PV *                              control mode bit:  0 CM
       setpoint (SP):        0                 0output limiting enabled bit:  0 OL
        process (PV):        0 *                       reset and gain range:  0 RG
        scaled error:        0 *                        scale setpoint flag:  0 SC
            deadband:        0                  loop update time too fast:  0 TF
        output (CV):         0 %*                derivitive (rate) action:  0 DA
                                               DB, set when error is in DB:  0 DB
         loop update:        0 [.01 secs]        output alarm, upper limit:  0 UL
                gain:        0 [/10]             output alarm, lower limit:  0 LL
               reset:        0 [/10 m/r]             setpoint out of range:  0 SP
                rate:        0 [/100 min]         process var out of range:  0 PV
          min scaled:        0                                 PID done:  0 DN
          max scaled:        0
  output (CV) limit:         NO *                          PID enabled:  0 EN
    output (CV) min:         0 %
    output (CV) max:         0 %
```

The left column in the display above lists further PID instruction parameters you must enter.

- **Auto/Manual AM (word 0, bit 1)** toggles between Auto and Manual. Auto indicates that the PID is controlling the output. (The bit is clear.) Manual indicates that the user is setting the output value. (The bit is set.) When tuning, we recommend that changes be made in the Manual mode, followed by a return to Auto. Output limiting is also applied in the Manual mode.
- **Mode TM (word 0, bit 0)** toggles values Timed and STI. Timed indicates that the PID updates its output at the rate specified in the loop update parameter.

Note

When using the timed mode, your processor scan time should be at least ten times faster than the loop update time to prevent timing inaccuracies or disturbances.

STI indicates that the PID updates its output every time it is scanned. When you select STI, the PID instruction should be programmed in an STI interrupt subroutine, and the STI routine should have a time interval equal to the setting of the PID "loop update" parameter. Set the STI period in word S:30. For example, if the loop update time contains the value 10 (for 100 ms), then the STI time interval must also equal 10 (for 10 ms).

- **Control CM (word 0, bit 2)** toggles values E=SP–PV and E=PV–SP. Direct acting (E=PV–SP) causes the output CV to increase when the input PV is larger than the setpoint SP (for example, a cooling application). Reverse acting (E=SP–PV) causes the output CV to increase when the input PV is smaller than the setpoint SP (for example, a heating application).

  – **Setpoint SP** (word 2) is the desired control point of the process variable. You can change this value with instructions in your ladder program. Write the value to the third word in the control block (for example write the value to N10:2 if your control block is N10:0). Without scaling, the range of this value is 0–16383. Otherwise, the range is minimum scaled (word 8) to maximum scaled (word 7).

  – **Gain K_c** (word 3) is the Proportional gain, ranging from 0.1 to 25.5. A rule of thumb is to set this gain to one half the value needed to cause the output to oscillate when the reset and rate terms (below) are set to zero.

    SLC 5/03 and higher processors – The valid range is 0 to 3276.7. ■

  – **Reset T_i** (word 4) is the Integral gain, ranging from 0.1 to 25.5 minutes per repeat. A rule of thumb is to set the reset time equal to the natural period measured in the above gain calibration.

    SLC 5/03 and higher processors – The valid range is 0 to 3276.7 minutes/repeat. Note that the value 1 will add the minimum integral term possible into the PID equation. ■

– **Rate T_d** (word 5) is the Derivative term. The adjustment range is 0.01 to 2.55 minutes. A rule of thumb is to set this value to 1/8 of the integral time above.

SLC 5/03 and higher processors – The valid range is 0 to 327.67 minutes. This word is not effected by the RG bit.

– **Maximum Scaled Smax** (word 7) – If the setpoint is to read in engineering units, then this parameter corresponds to the value of the setpoint in engineering units when the control input is 16383. Valid range is –16383 to +16383.

SLC 5/03 and higher processors – The valid range is –32768 to +32767.

– **Minimum Scaled Smin** (word 8) – If the setpoint is to read in engineering units, then this parameter corresponds to the value of the setpoint in engineering units when the control input is zero. Valid range is –16383 to +16383.

SLC 5/03 and higher processors – The valid range is –32768 to +32767.

Note

Smin – Smax *scaling allows you to enter the setpoint in engineering units. The deadband, error, and PV will be displayed in engineering units. The process variable, PV, is still be expected to be within the range of 0 to 16383. Use of* Smin – Smax *does not minimize PID PV resolution.*

SLC 5/03 and higher processors*: Scaled errors larger than +32767 or smaller than –32768 cannot be represented. If the scaled error is larger than +32767, it is represented as +32767. If the scaled error is smaller than –32768, it is represented as –32768.*

– **Deadband DB** (word 9) is a non-negative value. The deadband extends above and below the setpoint by the value you enter. The deadband is entered at the zero crossing of the process variable PV and the setpoint SP. This means that the deadband is in effect only after the process variable PV enters the deadband *and* passes through the setpoint SP. The valid range is 0 to scaled maximum, or 0 to 16383 when no scaling exists.

– **Loop Update** (word 13) is the time interval between PID calculations. The entry is in 0.01 second intervals. A rule of thumb is to enter a loop update time five to ten times faster than the natural period of the load (determined by setting the reset and rate parameters to zero and then increasing the gain until the output begins to oscillate). When in the STI mode, this value must equal the STI time interval value S:30. Valid range is 1 to 2.55 seconds.

SLC 5/03 and higher processors – The valid range is 0.01 to 10.24 seconds.

– **Scaled Process PV** (word 14) is for display only. This is the scaled value of the Process Variable (the analog input). Without scaling, the range of this value is 0–16383. Otherwise, the range is minimum scaled (word 8) to maximum scaled (word 7).

– **Scaled Error** (word 15) is for display only. This is the scaled error as selected by the control mode parameter. Range: scaled maximum to –scaled maximum, or 16383 to –16383 when no scaling exists.

Note

SLC 5/03 and higher processors specific*: Scaled errors larger than +32767 or smaller than –32768 cannot be represented. If the scaled error is larger than +32767, it is represented as +32767. If the scaled error is smaller than –32768, it is represented as –32768.*

– **Output CV%** (word 16**)** Displays the actual 0 to 16383 CV output in terms of percentage. (Range is 0 to 100%.) If you selected the AUTO mode with your programming software, this is for display only. If you selected manual mode and you are using your programming software's data monitor, you can change output CV% and the change is applied to CV. Writing to output CV% with your user program or a non-intelligent programming device does not affect the CV. When using a non-RSI programming software, you must write directly to CV, which ranges from 0 to 16383.

- **Output (CV) Limit OL (word 0, bit 3)** toggles between Yes and No. Select Yes if you want to limit the output to minimum and maximum values.

output CV%	YES (1) output CV% limiting selected	NO (0) output CV% limiting deselected
min	The value you enter is the minimum output percent that the control variable CV attain. If CV drops below this minimum value, the following occurs: • CV is set to the value you entered, and • The output alarm, lower limit LL bit is set.	The value you enter determines when the output alarm, lower limit bit is set. If CV drops below this minimum value, the output alarm, lower limit (LL) bit is set.
max	The value you enter is the maximum output percent that the control variable CV will attain. If CV exceeds this maximum value, the following occurs: • CV is set to the value you entered, and • The output alarm, upper limit UL bit is set.	The value you enter determines when the output alarm, upper limit bit is set. If CV exceeds this maximum value, the output alarm, upper limit (UL) bit is set.

PID Instruction Flags

```
          auto/manual: AUTO  *                          time mode Bit:  1 TM
                mode: STI   *                         auto/manual bit:  0 AM
             control: E=SP-PV *                       control mode bit:  0 CM
        setpoint (SP):     500                0output limiting enabled bit:  1 OL
         process (PV):       0 *                      reset and gain range:  0 RG
         scaled error:       0                         scale setpoint flag:  0 SC
             deadband:       5                    loop update time too fast:  0 TF
          output (CV):       0 %*                  derivitive (rate) action:  0 DA
                                                  DB, set when error is in DB:  0 DB
          loop update:     50 [.01 secs]           output alarm, upper limit:  0 UL
                gain:      25 [/10]                output alarm, lower limit:  0 LL
               reset:      10 [/10 m/r]                setpoint out of range:  0 SP
                rate:      1 [/100 min]             process var out of range:  0 PV
           min scaled:     0                                   PID done:  0 DN
           max scaled:     1000
    output (CV) limit:     NO *                             PID enabled:  0 EN
      output (CV) min:     0 %
      output (CV) max:     0 %
```

The right column of the display above shows various flags associated with the PID instruction. The following section describes those flags:

- **Time Mode Bit TM (word 0, bit 0)** specifies the PID mode. It is set when the TIMED mode is in effect. It is cleared when the STI mode is in effect. This bit can be set or cleared by instructions in your ladder program.
- **Auto/Manual Bit AM (word 0, bit 01)** specifies automatic operation when it is cleared and manual operation when it is set. This bit can be set or cleared by instructions in your ladder program.
- **Control Mode Bit CM (word 0, bit 02)** is cleared if the control is E=SP–PV. It is set if the control is E=PV–SP. This bit can be set or cleared by instructions in your ladder program.
- **Output Limiting Enabled Bit OL (word 0, bit 03)** is set when you have selected to limit the control variable using function key [F4]. This bit can be set or cleared by instructions in your ladder program.
- **SLC 5/03 and higher processors – Reset and Gain Range Enhancement Bit RG (word 0, bit 4)** When set, this bit causes the Reset Minute/Repeat value and the gain multiplier to be enhanced by a factor of 10, (reset multiplier of .01 and gain multiplier of .01).

  Example with bit 4 set: The Reset value of 1 indicates that the Integral value of 0.01 minutes/repeat (0.6 seconds/repeat) will be applied to the PID Integral algorithm. The gain value of 1 indicates that the error will be multiplied by 0.01 and applied to the PID algorithm.

When clear, this bit allows the Reset Minutes/Repeat value and the Gain multiplier value to be evaluated in the same units as the 5/02 PID instruction, (reset multiplier of 0.1 and gain multiplier of 0.1).

Example with bit 4 clear: The Reset value of 1 indicates that the Integral value of 0.1 minutes/repeat (6.0 seconds/repeat) will be applied to the PID Integral algorithm. The gain value of 1 indicates that the error will be multiplied by 0.1 and applied to the PID algorithm.

Note that the Rate multiplier is not affected by this selection. (The initial release software, version 4.0, may not allow you to enter this bit. However, you may alter the state of this bit directly in the control block.)

- **Scale Setpoint Flag SC (word 0, bit 05)** is cleared when setpoint scaling values are specified.
- **Loop Update Time Too Fast TF (word 0, bit 06)** is set by the PID algorithm if the loop update time you have specified cannot be achieved by the given program (because of scan time limitations).

  If this bit is set, try to correct the problem by updating your PID loop at a slower rate or move the PID instruction to an STI interrupt routine. Reset and rate gains will be in error if the instruction operates with this bit set.
- **Derivitive (Rate) Action Bit DA (word 0, bit 07)** When set, this bit causes the Derivitive (Rate) calculation to be evaluated on the Error instead of the PV. When clear, this bit allows the Derivitive (Rate) calculation to be evaluated the same as the 5/02 PID instruction, (where derivitive is performed on the PV). This bit is only used by SLC 5/03 and higher processors.
- **DB, Set When Error is in DB (word 0, bit 08)** is set when the process variable is within the 0 crossing deadband range.
- **Output Alarm, Upper Limit UL (word 0, bit 09)** is set when the calculated control output CV exceeds the upper CV limit.
- **Output Alarm, Lower Limit LL (word 0, bit 10)** is set when the calculated control output CV is less than the lower CV limit.
- **Setpoint Out of Range SP (word 0, bit 11)** is set when the setpoint exceeds the maximum scaled value or is less than the minimum scaled value.
- **Process Var Out of Range PV (word 0, bit 12)** is set when the unscaled (or raw) process variable exceeds 16383 or is less than zero.
- **PID Done DN (word 0, bit 13)** is set on scans where the PID algorithm is computed. It is computed at the loop update rate.
- **PID Enabled EN (word 0, bit 15)** is set while the rung of the PID instruction is enabled.

Control Block Layout

The control block length is fixed at 23 words and should be programmed as an integer file. PID instruction flags (word 0) and other parameters are located as follows:

Control Block Layout

15	14	13	12	11	10	09	08	07 ②	06	05	04 ②	03	02	01	00	Word
EN		DN	PV	SP	LL	UL	DB	DA	TF	SC	RG	OL	CM	AM	TM	0
* PID Sub Error Code (MSbyte)																1
* Setpoint SP																2
* Gain K_C																3
* Reset T_i																4
* Rate T_d																5
* Feed Forward Bias																6
* Setpoint Max (Smax)																7
* Setpoint Min (Smin)																8
* Deadband																9
INTERNAL USE DO NOT CHANGE																10
* Output Max																11
* Output Min																12
* Loop Update																13
Scaled Process Variable																14
Scaled Error SE																15
Output CV% (0–100%)																16
MSW Integral Sum								5/03 MSW Integral Sum								17
LSW Integral Sum								5/03 LSW Integral Sum								18
INTERNAL USE DO NOT CHANGE																19 20 21 22

OL, CM, AM, TM ①

① You may alter the state of these values with your ladder program.

② Applies to the SLC 5/03 and higher processors.

Do not alter the state of any PID control block value unless you fully understand its function and related effect on your process.

Runtime Errors

Error code 0036 appears in the status file when a PID instruction runtime error occurs. Code 0036 covers the following PID error conditions, each of which has been assigned a unique single byte code value that appears in the MSbyte of the second word of the control block.

Error Code	Description of Error Condition or Conditions		Corrective Action	
11H	*SLC 5/02*	*SLC 5/03 and higher*	*SLC 5/02*	*SLC 5/03 and higher*
	1) Loop update time **D_t** > 255, or 2) Loop update time **D_t** = 0	1) Loop update time **D_t** > 1024 2) Loop update time **D_t** = 0	Change loop update time **D_t** to 0 < **D_t** ≤ 255	Change loop update time **Dt** to 0 < **D_t** ≤ 1024
12H	*SLC 5/02*	*SLC 5/03 and higher*	*SLC 5/02*	*SLC 5/03 and higher*
	1) Proportional gain **K_c** > 255, or 2) Proportional gain **K_c** = 0	1) Proportional gain **K_c** < 0	Change proportional gain **K_c** to 0 < **K_c** ≤ 255	Change proportional gain **Kc** to 0 < **K_c**
13H	*SLC 5/02*	*SLC 5/03 and higher*	*SLC 5/02*	*SLC 5/03 and higher*
	Integral gain (reset) **T_i** > 255	Integral gain (reset) **T_i** < 0	Change integral gain (reset) **T_i** to 0 ≤ **T_i** ≤ 255	Change integral gain (reset) **Ti** to 0 ≤ **T_i**
14H	*SLC 5/02*	*SLC 5/03 and higher*	*SLC 5/02*	*SLC 5/03 and higher*
	Derivative gain (rate) **T_d** > 255	Derivative gain (rate) **T_d** < 0	Change derivative gain (rate) **T_d** to 0 ≤ **T_d** ≤ 255	Change derivative gain (rate) **T_d** to 0 ≤ **T_d**
21H (SLC 5/02 only)	1) Scaled setpoint max **Smax** > 16383, or 2) Scaled setpoint max **Smax** < –16383		Change scaled setpoint max **Smax** to –16383 ≤ **Smax** ≤ 16383	
22H (SLC 5/02 only)	1) Scaled setpoint min **Smin** > 16383, or 2) Scaled setpoint min **Smin** < –16383		Change scaled setpoint min **Smin** to –16383 ≤ **Smin** ≤ **Smax** ≤ 16383	
23H	Scaled setpoint min **Smin** > Scaled setpoint max **Smax**		Change scaled setpoint min **Smin** to –16383 ≤ **Smin** ≤ **Smax** ≤ 16383 (*SLC 5/03 and higher* –32768 to +32767)	

Error Code	Description of Error Condition or Conditions		Corrective Action	
31H	If you are using setpoint scaling and **Smin** > setpoint **SP** > **Smax**, or If you are not using setpoint scaling and 0 > setpoint **SP** > 16383, then during the initial execution of the PID loop, this error occurs and bit 11 of word 0 of the control block is set. However, during subsequent execution of the PID loop if an invalid loop setpoint is entered, the PID loop continues to execute using the old setpoint, and bit 11 of word 0 of the control block is set.		If you are using setpoint scaling, then change the setpoint **SP** to **Smin** ≤ **SP** ≤ **Smax**, or If you are not using setpoint scaling, then change the setpoint **SP** to 0 ≤ **SP** ≤ 16383.	
41H	Scaling Selected	Scaling Deselected	Scaling Selected	Scaling Deselected
	1) **Deadband** < 0, or 2) **Deadband** > **(Smax - Smin)**, or 3) **Deadband** > 16383 (5/02 specific)	1) **Deadband** < 0, or 2) **Deadband** > 16383	Change **deadband** to 0 ≤ **deadband** ≤ **(Smax – Smin)** ≤ 16383	Change **deadband** to 0 ≤ **deadband** ≤ 16383
51H	1) **Output high limit** < 0, or 2**) Output high limit** > 100		Change **output high limit** to 0 ≤ **output high limit** ≤ 100	
52H	1) **Output low limit** < 0, or 2**) Output low limit** > 100		Change **output low limit t**o 0 ≤ **output low limit** ≤ **output high limi**t ≤ 100	
53H	**Output low limi**t > **output high limit**		Change **output low limit** to 0 ≤ **output low limit** ≤ **output high limi**t ≤ 100	
60H	*SLC 5/02* – PID is being entered for the second time. (PID loop was interrupted by an I/O interrupt, which is then interrupted by the PID STI interrupt.)		You have at least three PID loops in your program: One in the main program or subroutine file, one in an I/O interrupt file, and one in the STI subroutine file. You must alter your ladder program and eliminate the potential nesting of PID loops.	

PID and Analog I/O Scaling

For the SLC 500 PID instruction, the numerical scale for both the process variable (PV) and the control variable (CV) is 0 to 16383. To use engineering units, such as PSI or degrees, you must first scale your analog I/O ranges within the above numerical scale. To do this, use the Scale (SCL) instruction and follow the steps described below.

1. Scale your analog input by calculating the slope (or rate) of the analog input range to the PV range (0 to 16383.) For example, an analog input with a range of 4 to 20mA has a decimal range of 3277 to 16384. The decimal range must be scaled across the range of 0 to 16383 for use as PV.

2. Scale the CV to span evenly across your analog output range. For example, an analog output which is scaled at 4 to 20mA has a decimal range of 6242 to 31208. In this case, 0 to 16383 must be scaled across the range of 6242 to 31208.

   Once you have scaled your analog I/O ranges to/from the PID instruction, you can enter the minimum and maximum engineering units that apply to your application. For example, if the 4 to 20mA analog input range represents 0 to 300 PSI, you can enter 0 and 300 as the minimum (Smin) and maximum (Smax) parameters respectively. The Process Variable, Error, Setpoint, and Deadband are displayed in engineering units in the PID Data Monitor screen. Setpoint and Deadband can be entered into the PID instruction using engineering units.

The following equations show the linear relationship between the input value and the resulting scaled value.

Scaled value = *(input value x slope) + offset*

Slope = *(scaled max. – scaled min.) / (input max. – input min)*

Offset = *scaled min. – (input min. x slope)*

Using the SCL Instruction

Use the following values in an SCL instruction to scale common analog input ranges to PID process variables.

Parameter	4 to 20mA	0 to 5V	0 to 10V
Rate/10,000	12,499	10,000	5,000
Offset	–4096	0	0

Use the following values in an SCL instruction to scale control variables to common analog outputs.

Parameter	4 to 20mA	0 to 5V	0 to 10V
Rate/10,000	15,239	10,000	19,999
Offset	6242	0	0

Using the SCP Instruction

Use the following values in an SCP instruction to scale your analog inputs to the PV range and scale the CV range to your analog output.

Parameter	4 to 20mA	0 to 5V	0 to 10V
Input minimum	0	0	0
Input maximum	16384	16384	32767
Scaled minimum	0	0	0
Scaled maximum	16383	16383	16383

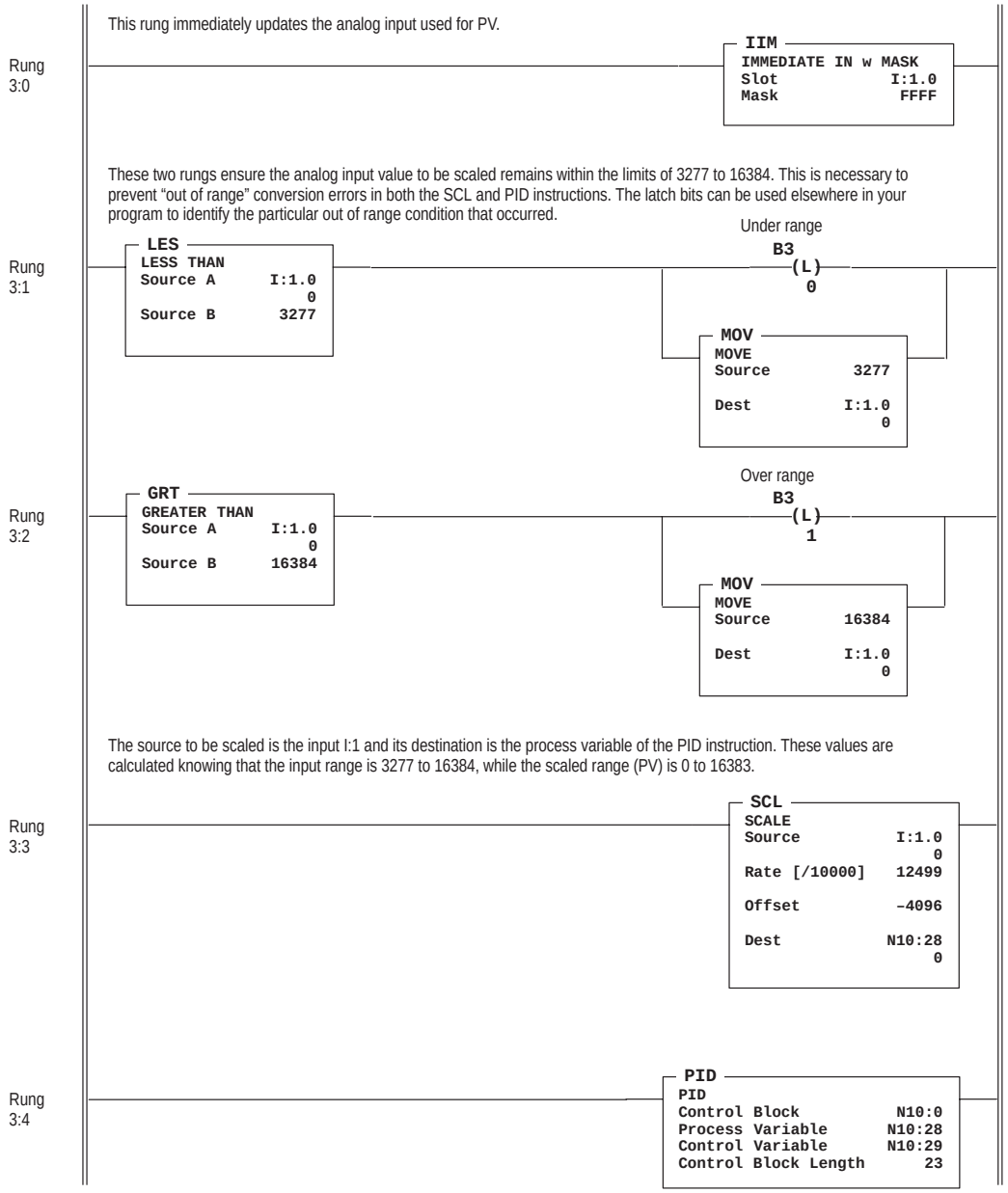
Use the following values in an SCP instruction to scale control variables to common analog outputs.

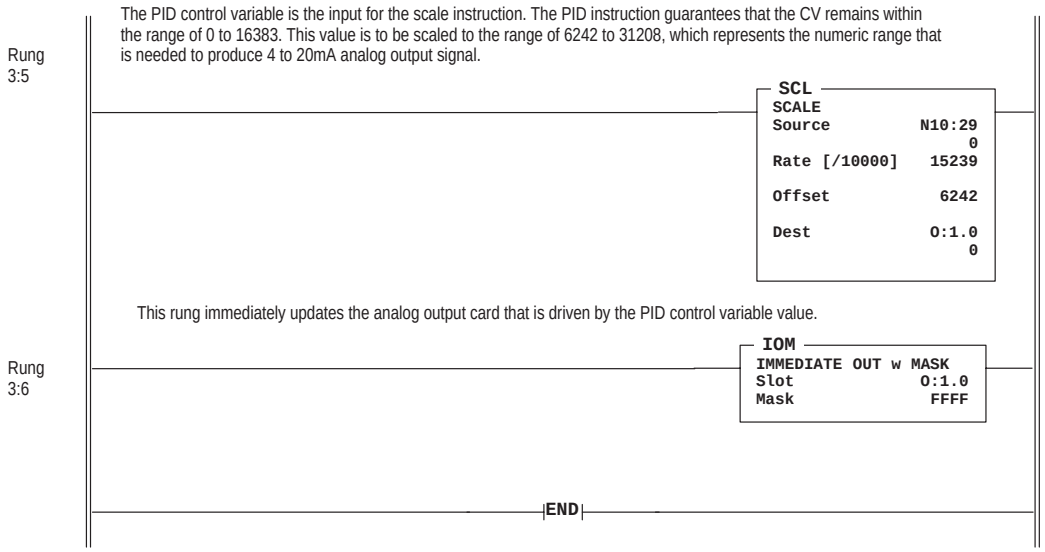
Parameter	4 to 20mA	0 to 5V	0 to 10V
Input minimum	3277	0	0
Input maximum	16383	16383	16383
Scaled minimum	6242	0	0
Scaled maximum	31208	16384	32764

Example

The following ladder diagram shows a typical PID loop that is programmed in the STI mode. This example is provided primarily to show the proper scaling techniques. It shows a 4 to 20mA analog input and a 4 to 20mA analog output. The following parameters are used:

- STI subroutine file (S:31) = 3
- STI Setpoint (S:30) = 10
- STI Enabled bit (S:2/1) = 1


This rung immediately updates the analog input used for PV.
Rung 3:0
IIM
IMMEDIATE IN w MASK
Slot I:1.0
Mask FFFF
These two rungs ensure the analog input value to be scaled remains within the limits of 3277 to 16384. This is necessary to prevent "out of range" conversion errors in both the SCL and PID instructions. The latch bits can be used elsewhere in your program to identify the particular out of range condition that occurred.
Under range
Rung 3:1
LES
LESS THAN
Source A I:1.0 0
Source B 3277
B3 0 (L)
MOV
MOVE
Source 3277
Dest I:1.0 0
Over range
Rung 3:2
GRT
GREATER THAN
Source A I:1.0 0
Source B 16384
B3 1 (L)
MOV
MOVE
Source 16384
Dest I:1.0 0
The source to be scaled is the input I:1 and its destination is the process variable of the PID instruction. These values are calculated knowing that the input range is 3277 to 16384, while the scaled range (PV) is 0 to 16383.
Rung 3:3
SCL
SCALE
Source I:1.0 0
Rate [/10000] 12499
Offset -4096
Dest N10:28 0
Rung 3:4
PID
PID
Control Block N10:0
Process Variable N10:28
Control Variable N10:29
Control Block Length 23



The STI routine should have a time interval equal to the setting of the PID “loop update” parameter.

Application Notes

The following paragraphs discuss:

- Input/Output Ranges
- Scaling to Engineering Units
- Zero-crossing Deadband
- Output Alarms
- Output Limiting with Anti-reset Windup
- The Manual Mode
- Feed Forward
- Time Proportioning Outputs

Input/Output Ranges

The input module measuring the process variable (PV) must have a full scale binary range of 0 to 16383. If this value is less than 0 (bit 15 set), then a value of zero is used for PV and the "Process var out of range" bit is set (bit 12 of word 0 in the control block). If the process variable is >16383 (bit 14 set), then a value of 16383 is used for PV and the "Process var out of range" bit is set.

The Control Variable, calculated by the PID instruction, has the same range of 0 to 16383. The Control Output (word 16 of the control block) has the range of 0 to 100%. You can set lower and upper limits for the instruction's calculated output values (where an upper limit of 100% corresponds to a Control Variable limit of 16383).

Scaling to Engineering Units

Scaling lets you enter the setpoint and zero-crossing deadband values in engineering units, and display the process variable and error values in the same engineering units. Remember, the process variable PV must still be within the range 0–16383. The PV is displayed in engineering units, however.

Select scaling as follows:

1. Enter the maximum and minimum scaling values Smax and Smin in the PID control block. Refer to the control block of the PID instruction on page 10–10. The Smin value corresponds to an analog value of zero for the lowest reading of the process variable, and Smax corresponds to an analog value of 16383 for the highest reading. These values reflect the process limits. Setpoint scaling is selected by entering a non-zero value for one or both parameters. If you enter the same value for both parameters, setpoint scaling is disabled.

For example, if measuring a full scale temperature range of – 73 (PV=0) to +1156° C (PV=16383), enter a value of –73 for Smin and 1156 for Smax. Remember that inputs to the PID instruction must be 0 to 16383. Signal conversions could be as follows:

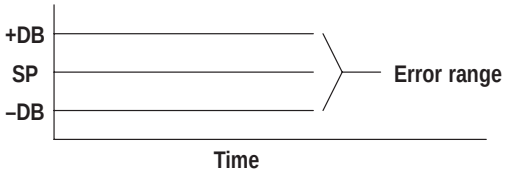
Process limits	–73 to +1156° C
Transmitter output (if used)	+4 to +20 mA
Output of analog input module	0 to 16383
PID instruction, Smin to Smax	–73 to +1156° C

2. Enter the setpoint (word 2) and deadband (word 9) in the same scaled engineering units. Read the scaled process variable and scaled error in these units as well. The control output percentage (word 16) is displayed as a percentage of the 0 to 16383 CV range. The actual value transferred to the CV output is always between 0 and 16383.

When you select scaling, the instruction scales the setpoint, deadband, process variable, and error. You must consider the effect on all these variables when you change scaling.

Zero–crossing Deadband DB

The adjustable deadband lets you select an error range above and below the setpoint where the output does not change as long as the error remains within this range. This lets you control how closely the process variable matches the setpoint without changing the output.



Zero-crossing is deadband control that lets the instruction use the error for computational purposes as the process variable crosses into the deadband until it crosses the setpoint. Once it crosses the setpoint (error crosses zero and changes sign) and as long as it remains in the deadband, the instruction considers the error value zero for computational purposes.

Select deadband by entering a value in the deadband storage word (word 9) in the control block. The deadband extends above and below the setpoint by the value you enter. A value of zero inhibits this feature. The deadband has the same scaled units as the setpoint if you choose scaling.

Output Alarms

You may set an output alarm on the control output (CO) at a selected value above and/or below a selected output percent. When the instruction detects that the output (CO) has exceeded either value, it sets an alarm bit (bit 10 for lower limit, bit 9 for upper limit) in word 0 of the PID control block. Alarm bits are reset by the instruction when the output (CO) comes back inside the limits. The instruction does not prevent the output (CO) from exceeding the alarm values unless you select output limiting.

Select upper and lower output alarms by entering a value for the upper alarm (word 11) and lower alarm (word 12). Alarm values are specified as a percentage of the output. If you do not want alarms, enter zero and 100% respectively for lower and upper alarm values and ignore the alarm bits.

Output Limiting with Anti-Reset Windup

You may set an output limit (percent of output) on the control output. When the instruction detects that the output (CO) has exceeded a limit, it sets an alarm bit (bit 10 for lower limit, bit 9 for upper limit) in word 0 of the PID control block, and prevents the output (CO) from exceeding either limit value. The instruction limits the output (CO) to 0 and 100% if you choose not to limit.

Select upper and lower output limits by setting the limit enable bit (bit 3 of control word 0), and entering an upper limit (word 11) and lower limit (word 12). Limit values are a percentage (0 to 100%) of the control output (CO).

The difference between selecting output alarms and output limits is that you must select output limiting to enable limiting. Limit and alarm values are stored in the same words. Entering these values enables the alarms, but not limiting. Entering these values and setting the limit enable bit enables limiting and alarms.

Anti-reset windup is a feature that prevents the integral term from becoming excessive when the output (CO) reaches a limit. When the sum of the PID and bias terms in the output (CO) reaches the limit, the instruction stops calculating the integral sum until the output (CO) comes back in range. The integral sum is contained in words 17 and 18 of the control block.

The Manual Mode

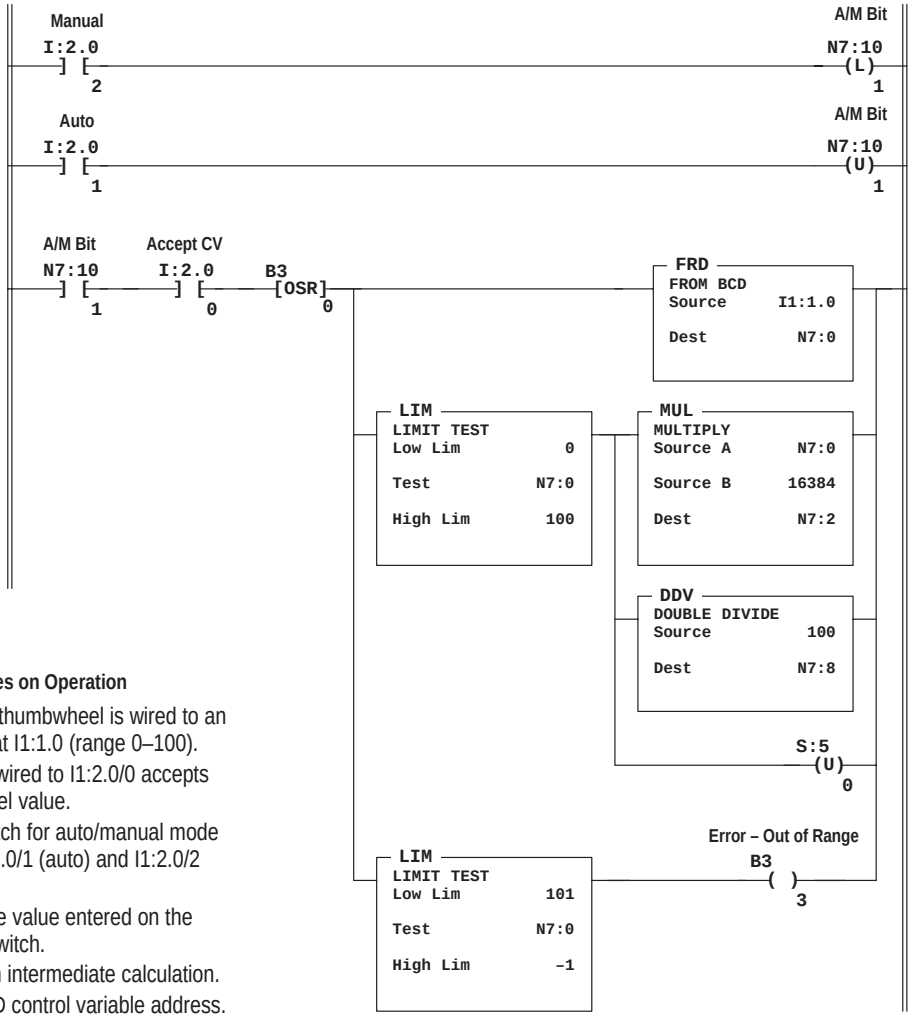
In the manual mode, the PID algorithm does not compute the value of the control variable. Rather, it uses the value as an input to adjust the integral sum (words 17 and 18) so that a bumpless transfer takes place upon re-entering the AUTO mode.

In the manual mode, the programmer allows you to enter a new CV value from 0 to 100%. This value is converted into a number from 0 to 16383 and written to the Control Variable address. If you are using an analog output module for this address, you must save (compile) the program with the File Protection option set to None. This allows writing to the output data table. If you do not perform this save operation, you are not able to set the output level in the manual mode. If your ladder program sets the manual output level, design your ladder program to write to the CV address when in the manual mode. Note that this number is in the range of 0 to 16383, not 0 to 100. Writing to the CV percent (word 16) with your ladder program has no effect in the manual mode.

The example on the next page shows how you can manually control the control variable (CV) output with your ladder program.

PID Rungstate

If the PID rung is false, the integral sum (words 17 and 18) is cleared and CV remains in its last state.



Notes on Operation

A 3-digit BCD thumbwheel is wired to an input module at I1:1.0 (range 0–100).

A pushbutton wired to I1:2.0/0 accepts the thumbwheel value.

A selector switch for auto/manual mode is wired to I1:2.0/1 (auto) and I1:2.0/2 (manual).

N7:0 stores the value entered on the thumbwheel switch.

N7:2 stores an intermediate calculation.

N7:8 is the PID control variable address.

N7:10 is the control block address of the PID instruction.

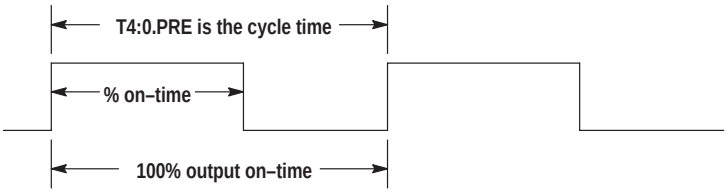
N7:26 Percent output is updated automatically by the PID instruction.

Feed Forward or Bias

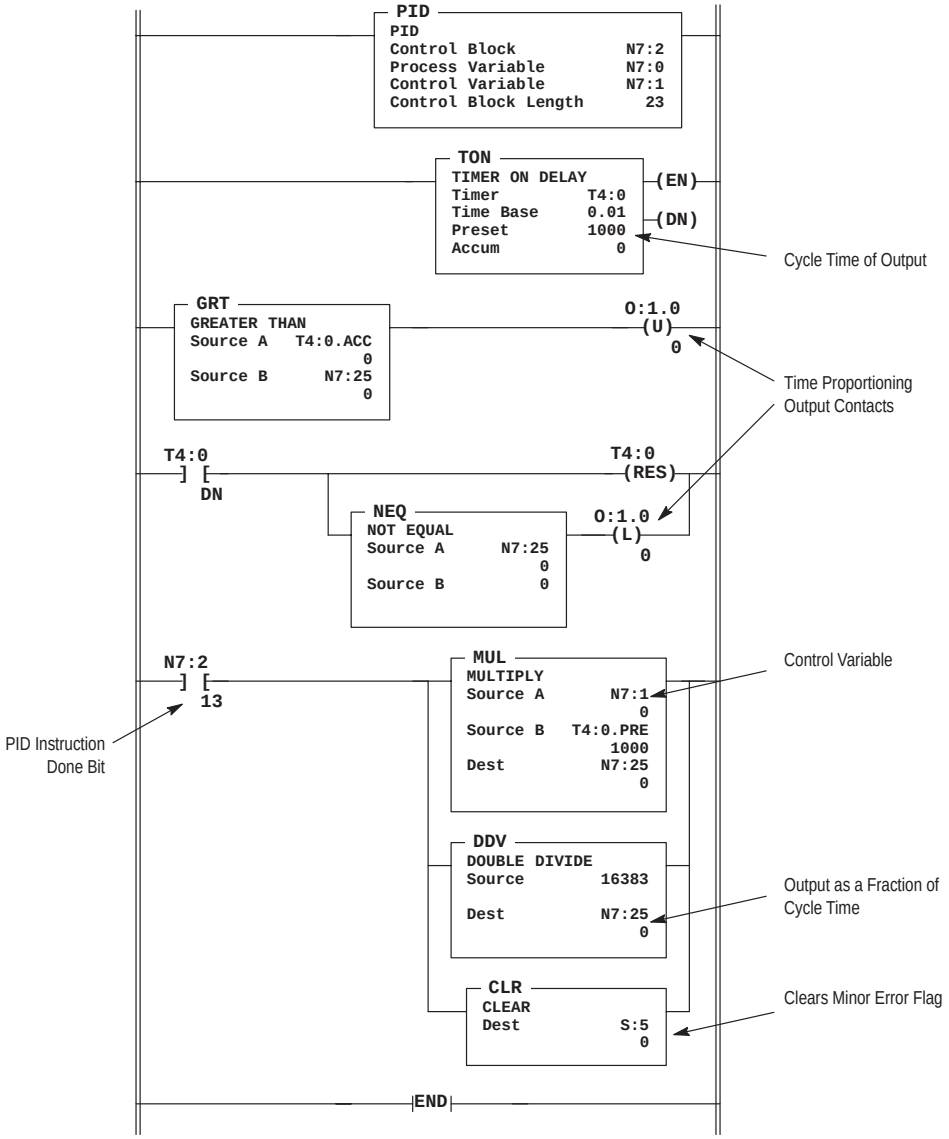
Applications involving transport lags may require that a bias be added to the CV output in anticipation of a disturbance. This bias can be accomplished using the processor by writing a value to the Feed Forward Bias element, the seventh element (word 6) in the control block file. (See page 10–10.) The value you write is added to the output, allowing a feed forward action to take place. You may add a bias by writing a value between –16383 and +16383 to word 6 with your programming terminal or ladder program.

Time Proportioning Outputs

For heating or cooling applications, the Control Variable analog output is typically converted to a time-proportioning output. While this cannot be done directly with the processor, you can use the program on the following page to convert the Control Variable to a time proportioning output. In this program, cycle time is the preset of timer T4:0. Cycle time relates to % on-time as follows:



Example – Time proportioning outputs



PID Tuning

PID tuning requires a knowledge of process control. If you are inexperienced, it will be helpful if you obtain training on the process control theory and methods used by your company.

There are a number of techniques that can be used to tune a PID loop. The following PID tuning method is general, and is limited in terms of handling load disturbances. When tuning, we recommend that changes be made in the MANUAL mode, followed by a return to AUTO. Output limiting is applied in the MANUAL mode.

Note *This method requires that the PID instruction controls a non-critical application in terms of personal safety and equipment damage.*

Note *The PID tuning procedure may not work for all cases. It is strongly recommended to use a PID Loop tuner package for the best result (ex. RSTune, Catalog #9323-1003D, PID Logistic for PLC 500 A.I Series or P/N 9323-S5250D or PID Logistic for PLC 500 A.I Series Stand Alone Package P/N 9323–S5255D).*

Procedure

1. Create your ladder program. Make certain that you have properly scaled your analog input to the range of the process variable PV and that you have properly scaled your control variable CV to your analog output.

2. Connect your process control equipment to your analog modules. Download your program to the processor. Leave the processor in the program mode.

Ensure that all possibilities of machine motion have been considered with respect to personal safety and equipment damage. It is possible that your output CV may swing between 0 and 100% while tuning.

Note *If you want to verify the scaling of your continuous system and/or determine the initial loop update time of your system, go to the procedure on page 10–26.*

3. Enter the following values: the initial setpoint SP value, a reset T_i of 0, a rate T_d of 0, a gain K_c of 1, and a loop update of 5.

   Set the PID mode to STI or Timed, per your ladder diagram. If STI is selected, ensure that the loop update time equals the STI time interval.

   Enter the optional settings that apply (output limiting, output alarm, Smax – Smin scaling, feedforward).

4. Get prepared to chart the CV, PV, analog input, or analog output as it varies with time with respect to the setpoint SP value.

5. Place the PID instruction in the MANUAL mode, then place the processor in the Run mode.

6. While monitoring the PID display, adjust the process manually by writing to the CO percent value.

7. When you feel that you have the process under control manually, place the PID instruction in the AUTO mode.

8. Adjust the gain while observing the relationship of the output to the setpoint over time.

When using the SLC 5/02 processor, gain adjustments disrupt the process when you change values. To avoid this disruption, switch to the MANUAL mode prior to making your gain change, then switch back to the AUTO mode. When using an SLC 5/03 or higher processor, gain changes do not disrupt the process; therefore, you do not need to switch to the MANUAL mode.

9. When you notice that the process is oscillating above and below the setpoint in an even manner, record the time of 1 cycle. That is, obtain the natural period of the process.

Natural Period ≅ 4x deadtime

Record the gain value. Return to the MANUAL mode (stop the process if necessary).

10. Set the loop update time (and STI time interval if applicable) to a value of 5 to 10 times faster than the natural period.

For example, if the cycle time is 20 seconds, and you choose to set the loop update time to 10 times faster than the natural rate, set the loop update time to 200, which would result in a 2-second rate.

11. Set the gain K_c value to 1/2 the gain needed to obtain the natural period of the process. For example, if the gain value recorded in step 9 was 80, set the gain to 40.

12. Set the reset term T_i to approximate the natural period. If the natural period is 20 seconds, as in our example, you would set the reset term to 3 (0.3 minutes per repeat approximates 20 seconds).

13. Now set the rate T_d equal to a value 1/8 that of the reset term. For our example, the value 4 will be used to provide a rate term of 0.04 minutes per repeat.

14. Place the process in the AUTO mode. If you have an ideal process, the PID tuning will be complete.

15. To make adjustments from this point, place the PID instruction in the MANUAL mode, enter the adjustment, then place the PID instruction back in the AUTO mode.

This technique of going to MANUAL, then back to AUTO ensures that most of the "gain error" is removed at the time each adjustment is made. This allows you to see the effects of each adjustment immediately. Toggling the PID rung allows the PID instruction to restart itself, eliminating all of the "integral buildup." You may want to toggle the PID rung false while tuning to eliminate the effects of previous tuning adjustments.

Verifying the Scaling of Your Continuous System

To ensure that your process is linear, and that your equipment is properly connected and scaled, do the following:

1. Place the PID instruction in manual and enter the following parameters:

- type: `0` for Smin
- type: `100` for Smax
- type: `0` for CO%

2. Enter the REM Run mode and verify that PV=0.

3. Type: `20` in CO%

4. Record the PV = _____

5. Type: `40` in CO%.

6. Record the PV = _____

7. Type: `60` in CO%.

8. Record the PV = _____

9. Type: `80` in CO%.

10. Record the PV = _____

11. The values you recorded should be offset from CO% by the same amount. This proves the linearity of your process. The following example shows an offset progression of fifteen.

CO 20% = PV 35%

CO 40% = PV 55%

CO 60% = PV 75%

CO 80% = PV 95%

If the values you recorded are not offset by the same amount:

- Either your scaling is incorrect, or
- the process is not linear, or
- your equipment is not properly connected and/or configured.

Make the necessary corrections and repeat steps 2–10.

Determining the Initial Loop Update Time

To determine the approximate loop update time that should be used for your process, perform the following:

1. Place the normal application values in Smin and Smax.

2. Type: `50` in CO%.

3. Type: `60` in CO% and immediately start your stopwatch.

4. Watch the PV. When the PV starts to change, stop your stopwatch. Record this value. It is the deadtime.

5. Multiply the deadtime by 4. This value approximates the natural period. For example, if:

deadtime = 3 seconds, then 4 $\times$ 3 = 12 seconds ($\cong$ natural period)

6. Divide the value obtained in step 5 by 10. Use this value as the loop updated time. For example, if:

natural period = 12 seconds, then 12 $\div$ 10 = 1.2 seconds.

Therefore, the value 120 would be entered as the loop update time.
(120 $\times$ 10 ms = 1.2 seconds)

7. Enter the following values: the initial setpoint SP value, a reset T_i of 0, a rate T_d of 0, a gain K_c of 1, and the loop update time determined in step 17.

Set the PID mode to STI or Timed, per your ladder diagram. If STI is selected, ensure that the loop update time equals the STI time interval.

Enter the optional settings that apply (output limiting, output alarm, Smax – Smin scaling, feedforward).

8. Return to page 10–25 and complete the tuning procedure starting with step 4.

11 ASCII Instructions

This chapter contains general information about the ASCII instructions and explains how they function in your application program. Each of the instructions includes information on:

- what the instruction symbol looks like
- how to use the instruction

ASCII Instructions

Instruction		Purpose	Page
Mnemonic	**Name**		
ABL	Test Buffer for Line	Determine the number of characters in the buffer, up to and including the end of line character.	**11–6**
ACB	Number of Characters in Buffer	Determine the total number of characters in the buffer.	**11–7**
ACI	String to Integer	Convert a string to an integer value.	**11–9**
ACL	ASCII Clear Receive and/or Send Buffer	Clear the receive and/or transmit buffers.	**11–10**
ACN	String Concatenate	Link two strings into one.	**11–11**
AEX	String Extract	Extract a portion of a string to create a new string.	**11–12**
AHL	ASCII Handshake Lines	Set or reset modem handshake lines.	**11–13**
AIC	Integer to String	Convert an integer value to a string.	**11–14**
ARD	ASCII Read Characters	Read characters from the input buffer and place them into a string.	**11–15**

continued on next page

Instruction		Purpose	Page
Mnemonic	Name		
ARL	ASCII Read Line	Read one line of characters from the input buffer and place them into a string.	**11–18**
ASC	String Search	Search a string.	**11–20**
ASR	ASCII String Compare	Compare two strings.	**11–21**
AWA	ASCII Write with Append	Write a string with user-configured characters appended.	**11–22**
AWT	ASCII Write	Write a string.	**11–25**

ASCII Instruction Overview

ASCII instructions are available in SLC 5/03 OS301 and above processors, and all SLC 5/04 and SLC 5/05 processors. There are two types of ASCII instructions:

- ASCII port control – these include instructions that use or alter the communication channel for receiving or transmitting data. When using these instructions, the system configuration must be set to "User mode."

  (ABL, ACB, ACL, AHL*, ARD, ARL, AWA*, AWT*)
  *must be in user or system mode

  ASCII port control instructions are queued in the order that they are executed and are dependent on one another to execute (except ACL which executes immediately). For example, if you have an ARD (ASCII Read instruction) and then an AWT (ASCII Write instruction), the Done bit or the Error bit of the ARD must be set before the AWT can begin executing (even if the AWT was enabled while the processor was executing the ARD). A second ASCII port control instruction cannot begin executing until the first has completed. However, the processor *does not* wait for an ASCII port control instruction to complete before continuing to execute your ladder program.

- ASCII string control – these include instructions that manipulate string data. (ACI, ACN, AEX, AIC, ASC, ASR)

  ASCII string control instructions execute immediately. They are never sent to the queue to wait their turn for execution.

Protocol Parameter Overview

Listed below are the ASCII protocol parameters that you set via the Channel 0 configuration screens in your programming software.

Description	Specification
Baud Rate	Toggles between 110, 300, 600, 1.2K, 2.4K, 4.8K, 9.6K, and 19.2K (additional rate of 38.4K for SLC 5/05 only). The default is 1.2K (19.2K for SLC 5/05 only).
Start Bits	The default is 1 and cannot be changed.
Stop Bits	Options include 1, 1.5, and 2 The default is 1.
Parity	Toggles between None, Odd, and Even. The default is None.
Data Bits	Toggles between 7 and 8. The default is 8.
Termination Characters	Allows you to configure up to 2 ASCII characters. The default is CR.
Append Characters	Allows you to configure up to 2 ASCII characters. The AWA instruction adds the characters to the end of every string to serve as termination characters for the receiving device. The default is CR LF.

Using the ASCII Data File Type

These are 1-word elements. Assign ASCII addresses as follows:

Format	Explanation		
Af:e/b	**A**	**ASCII file**	
	f	**File number. A file number between 9- 255 can be used.**	
	:	**Element delimiter**	
	e	**Element number**	**Ranges from 0- 255. This is a 1-word element.**
	/	**Bit delimiter**	
	b	**Bit number**	**Bit location within the element. Ranges from 0- 15.**

Examples:

A9:2	Element 2, ASCII file 9
A10:0/7	Bit 7, Element 0, ASCII file 10

Using the String (ST) Data File Type

This file type is valid for SLC 5/03 OS301 and higher, SLC 5/04, and SLC 5/05 processors. These are 42-word elements. You can address string lengths by adding a .LEN to any string address (for example, ST17:1.LEN). Valid string data file numbers are 9–255.

String lengths must be between 0 and 82. In general, lengths that are outside of this range cause the processor to set the ASCII Error bit (S:5/15) and the instruction is not executed.

Note

You configure append or end-of-line characters via the Channel Configuration screen. The default append characters are carriage return and line feed; the default end-of-line (termination) character is a carriage return.

All instructions except ACL and AHL will error if the port is disabled.

Assign string addresses as follows:

Format	Explanation		
STf:e.s/b	ST	String file	
	f	File number. A file number between 9- 255 can be used.	
	:	Element delimiter	
	e	Element number	Ranges from 0- 255. These are 42-word elements. 16 bits per element.
	.	Subelement delimiter	
	s	Subelement number	Ranges from 0 - 41. Word 0 is the length, .LEN.
	/	Bit delimiter	
	b	Bit number	Bit location within the element. Ranges from 0- 15. Bit level addressing is not available for Word 0 of string elements.

Examples:

ST9:2	Element 2, string file 9
ST10:2.3/8	Bit 8 in subelement 3 of element 2, string file 10

Entering Parameters

The control element for ASCII instructions includes eight status bits, an error code byte, and two character words:

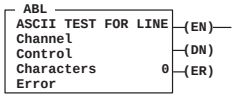
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Word 0	EN	EU	DN	EM	ER	UL	IN	FD	Error Code							
Word 1	Number of characters for sending or receiving (LEN)															
Word 2	Number of characters sent or received (POS)															

EN = Enable Bit
EU = Queue Bit
DN = Asynchronous Done Bit
EM = Synchronous Done Bit
ER = Error Bit
UL = Unload Bit
IN = Running Bit (This bit is the IN bit in the control data file [R6:].)
FD = Found Bit

- **Found Bit FD** (bit 8) indicates that the instruction found the end of characters or termination characters in the buffer (applies to ABL and ACB instructions)
- **Running Bit IN** (bit 9) indicates that a queued instruction is executing.
- **Unload Bit UL** (bit 10) ceases instruction operation before (may be queued) or during execution. If this bit is set while an instruction is executing, any data already processed is sent to the destination. Note that the instruction is not removed from the queue; any remaining data is just not processed. You set this bit.
- **Error Bit ER** (bit 11) indicates that an error occurred while executing the instruction, such as a mode change via channel 1, or the instruction was cancelled using the UL bit or ACL instruction.
- **Synchronous Done Bit EM** (bit 12) is set concurrently to a program scan to indicate the completion of an ASCII instruction.
- **Asynchronous Done Bit DN** (bit 13) is set opposite to a program scan when an instruction successfully completes its operation. Note that an instruction can take longer than one program scan to finish executing.
- **Queue Bit EU** (bit 14) indicates that an ASCII instruction was placed in the ASCII queue. This action is delayed if the queue is already filled. The queue may contain up to 16 instructions.
- **Enable Bit EN** (bit 15) indicates that an instruction is enabled due to a false-to-true transition. This bit remains set until the instruction has completed executing or errors.

Test Buffer for Line (ABL)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

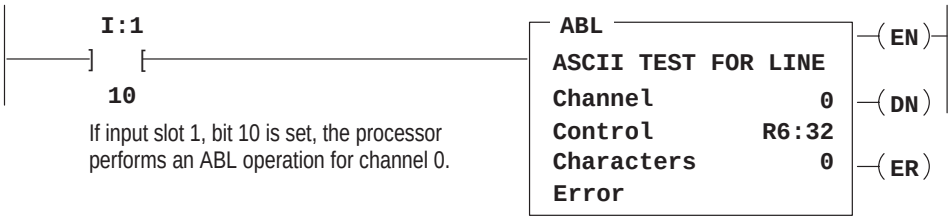
Use the ABL instruction to determine the total number of characters in the input buffer, up to and including the end-of-line characters (termination). This instruction looks for two termination characters that you configure via the ASCII port configuration screen. On a false-to-true transition, the processor reports the number of characters in the POS field of the ASCII control block. The serial port must be configured for User mode.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **Control** is the area that stores the control register required to operate the instruction.
- **Characters** are the number of characters in the buffer that the processor finds (0–1024). This parameter is display only and resides in word 2 of the control block.
- **Error** displays the hexadecimal error code that indicates why the ER bit was set in the control data file (R6:). See page 11–27 for error code descriptions.

Example



If input slot 1, bit 10 is set, the processor performs an ABL operation for channel 0.

When the rung goes from false-to-true, the Enable bit (EN) is set. The instruction is put in the ASCII instruction queue, the Queue bit (EU) is set, and program scan continues. The instruction is then executed outside of the program scan. However, if the queue is empty the instruction executes immediately. Upon execution, the Run bit (RN) is set.

The processor determines the number of characters (up to and including the end-of-line/termination characters) and puts this value in the position field. The Done bit (DN) is then set.

If a zero appears in the POS field, no end-of-line/termination characters were found. The Found bit (FD) is set if the position field was set to a non-zero value.

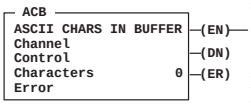
When the program scans the instruction and finds the Done bit (DN) set, the processor then sets the Synchronous Done bit (EM). The EM bit acts as a secondary done bit corresponding to the program scan.

The Error bit (ER) is set during the execution of the instruction if:

- the instruction is aborted – serial port not in User mode
- the instruction is aborted due to channel mode change
- the Unload bit (UL) is set and the instruction is not executed

Number of Characters In Buffer (ACB)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

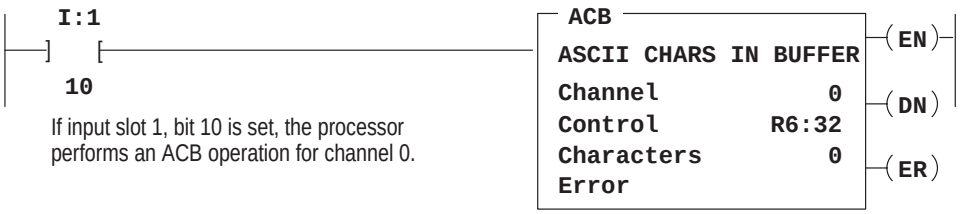
Use the ACB instruction to determine the total characters in the buffer. On a false-to-true transition, the processor determines the total number of characters and records it in the position field of the ASCII control block. The serial port must be in User mode.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **Control** is the area that stores the control register required to operate the instruction.
- **Characters** are the number of characters in the buffer that the processor finds (0–1024). This parameter is display only.
- **Error** displays the hexadecimal error code that indicates why the ER bit was set in the control data file (R6:). See page 11–27 for error descriptions.

Example



When the rung goes from false-to-true, the Enable bit (EN) is set. When the instruction is placed in the ASCII queue, the Queue bit (EU) is set. The Running bit (RN) is set when the instruction is executing. The Done bit (DN) is set on completion of the instruction.

The processor determines the number of characters in the buffer and puts this value in the position field of the control block. The Done bit (DN) is then set. If a zero appears in the characters field, no characters were found.

When the program scans the instruction and finds the Done bit (DN) set, the processor then sets the Synchronous bit (EM). The EM bit acts as a secondary done bit corresponding to the program scan.

The Error bit (ER) is set during the execution of the instruction if:

- the instruction is aborted – serial port not in User mode
- the instruction is aborted due to channel mode change
- the Unload bit (UL) is set and the instruction is not executed

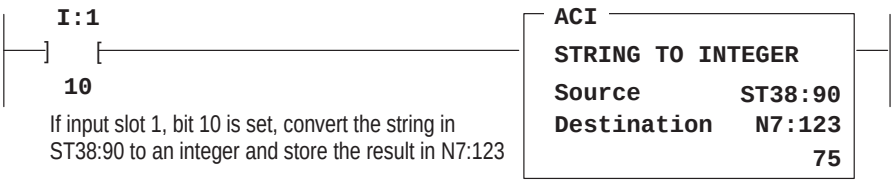
String to Integer (ACI)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

Output Instruction

Use the ACI instruction to convert an ASCII string to an integer value between –32,768 and 32,767.

Example



If input slot 1, bit 10 is set, convert the string in ST38:90 to an integer and store the result in N7:123

The processor searches the source (file type **ST**) for the first character between 0 and 9. All numeric characters are extracted until a non-numeric character or the end of the string is reached. Action is taken *only* if numeric characters are found. If the string contains an invalid length (<0 or >82) the ASCII Error bit S:5/15 is set. Commas and signs (+, –) are allowed in the string. However, only the minus sign is displayed in the data table.

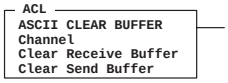
The extracted numeric string is then converted to an integer. The ASCII Error bit S:5/15 is set if the string contains an invalid string length. The value of 32,767 is returned as the result.

This instruction also sets the arithmetic flags (found in word 0, bits 0–3 in the processor status file S:0):

With this Bit:		The Processor:
S:0/0	Carry (C)	is reserved.
S:0/1	Overflow (V)	sets if the integer value is outside of the valid range.
S:0/2	Zero (Z)	sets if the integer value is zero.
S:0/3	Sign (S)	sets if the result is negative.

ASCII Clear Receive and/or Send Buffer (ACL)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓



Output Instruction

Use this instruction to clear an ASCII buffer. ASCII instructions are removed from the queue and then the Error bit (ER) is set. This instruction executes immediately upon the rung transitioning to a true state. The instruction works when the channel is in User Mode or System Mode. In System Mode, only clearing the send buffer will operate and then only if DF1 is selected as the System Mode protocol.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **Clear Receive Buffer** clears the receive buffer and removes the ARD and ARL instructions from the queue. The Error bit (ER) is set in each of these instructions.
- **Clear Send Buffer** clears the send buffer and removes the AWA and AWT instructions from the queue. The Error bit (ER) is set in each of these instructions.

When Clear Receive Buffer and Clear Send Buffer are both set to Yes, all instructions are removed from the queue.

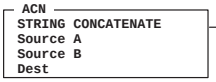
Example

If input slot 1, bit 10 is set, then only clear the receive buffer for channel 0.

When the rung goes true, the selected buffer(s) will be cleared and the ASCII instruction(s) are removed from the ASCII instruction queue.

String Concatenate (ACN)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓



The ACN instruction combines two strings using ASCII strings as operands. The second string is appended to the first and the result stored in the destination.

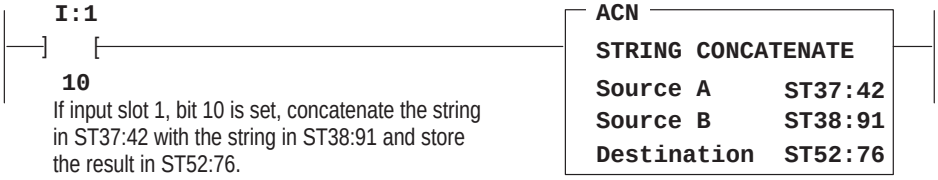
Output Instruction

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source A** is the first string in the concatenation procedure.
- **Source B** is the second string in the concatenation procedure.
- **Destination** is where the result of Source A and B is stored.

Example

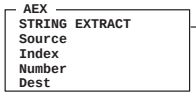


If input slot 1, bit 10 is set, concatenate the string in ST37:42 with the string in ST38:91 and store the result in ST52:76.

Only the first 82 characters (0 – 81) are written to the destination.
If the result is < 0 or > 82 the ASCII Error bit S:5/15 is set.

String Extract (AEX)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

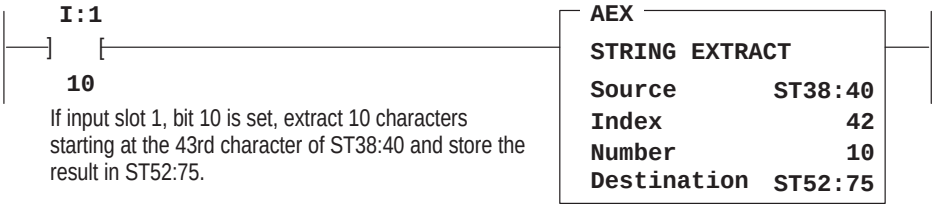
Use the AEX instruction to create a new string by taking a portion of an existing string and linking it to a new string.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is the existing string. The source value is not affected by this instruction.
- **Index** is the starting position (from 1 to 82) of the string you want to extract. (An index of 1 indicates the left-most character of the string.)
- **Number** is the number of characters (from 1 to 82) you want to extract, starting at the indexed position. If the index plus the number is greater than the total characters in the source string, the destination string will be the characters from the index to the end of the source string.
- **Destination** is the string element (ST) where you want the extracted string stored.

Example



If input slot 1, bit 10 is set, extract 10 characters starting at the 43rd character of ST38:40 and store the result in ST52:75.

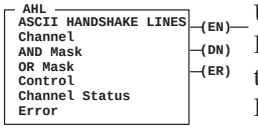
The following conditions cause the processor to set the ASCII Error bit (S:5/15):

- invalid source string length or string length of zero
- index or number values outside of range
- index value greater than the length of the source string

The destination string is not changed in any of the above error conditions. However, the destination will be changed if the index value plus the number value are greater than the string length. Note that the ASCII Error bit (S:5/15) is not set.

ASCII Handshake Lines (AHL)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

Use the AHL instruction to set or reset the RS-232 Data Terminal Ready (DTR) and Request to Send (RTS) handshake control lines for your modem. On a false-to-true transition, the processor uses the two masks to determine whether to set or reset the DTR and RTS lines, or leave them unchanged. This instruction will operate when the port is in either mode or is disabled.

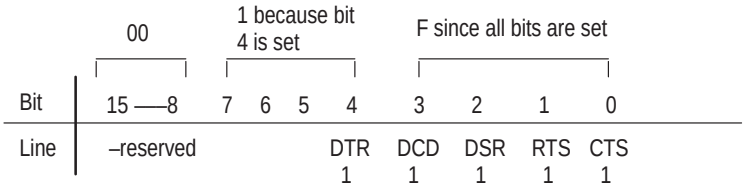
Note *Make sure the automatic modem control used by the port does not conflict with this instruction.*

Entering Parameters

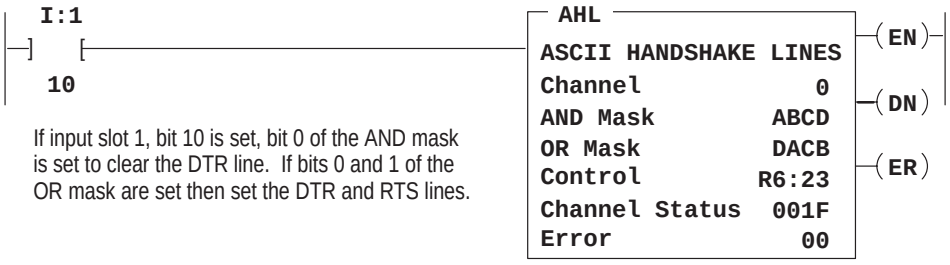
Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **AND Mask** is the type of mask used to reset the DTR and RTS control lines. Bit 0 corresponds to the DTR line and bit 1 corresponds to the RTS control line. A 1 at the mask bit causes the line to be reset; a 0 leaves the line unchanged. Note that mask values do not have a one-to-one correspondence to the modem control lines.
- **OR Mask** is the type of mask used to set the DTR and RTS control lines. Bit 0 corresponds to the DTR line and bit 1 corresponds to the RTS control line. A 1 at the mask bit causes the line to be set; a 0 leaves the line unchanged. Note that mask values do not have a one-to-one correspondence to the modem control lines.
- **Control** is the area that stores the control register required to operate the instruction.
- **Channel Status** displays the current status (0000 to 001F) of the handshake lines for the channel, specified above. This field is display only and resides in word 2 of the control element.
- **Error** displays the hexadecimal error code that indicates why the ER bit was set in the control data file (R6:). See page 11–27 for error code descriptions.

Example: The following shows the channel status as 001F.



Example

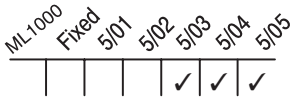


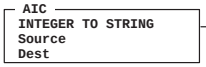
If input slot 1, bit 10 is set, bit 0 of the AND mask is set to clear the DTR line. If bits 0 and 1 of the OR mask are set then set the DTR and RTS lines.

The Error bit (ER) is set during the execution of the instruction if:

- the instruction is aborted due to channel mode change
- the Unload bit (UL) is set and the instruction is not executed

Integer to String (AIC)

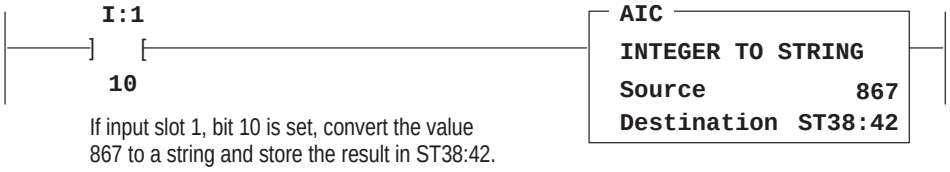




Output Instruction

The AIC instruction converts an integer value (–32,768 and 32,767) to an ASCII string. The source can be a constant or an integer address.

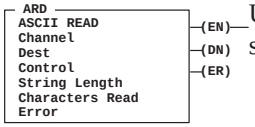
Example



If input slot 1, bit 10 is set, convert the value 867 to a string and store the result in ST38:42.

ASCII Read Characters (ARD)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓



Output Instruction

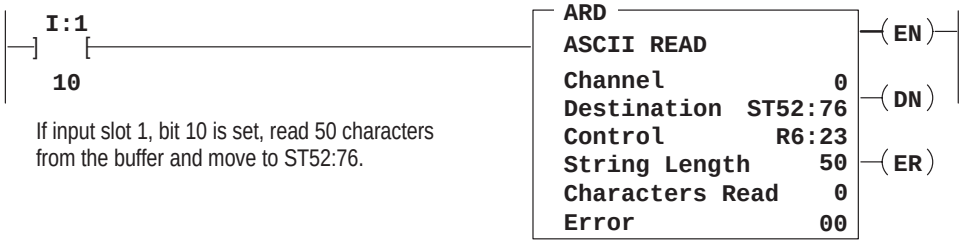
Use the ARD instruction to read characters from the buffer and store them in a string. To repeat the operation, the rung must go from false-to-true.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **Destination** is the string element where you want the characters stored.
- **Control** is the address of the control block used to store data for the ARD instruction.
- **String Length** (.LEN) is the number of characters you want to read from the buffer. The maximum is 82 characters. If you specify a length larger than 82, only the first 82 characters will be read. (A 0 defaults to 82.) This is word 1 in the control block.
- **Characters Read** (.POS) are the number of characters that the processor moved from the buffer to the string (0 to 82). This field is updated during the execution of the instruction and is display only. This is word 2 in the control block.
- **Error** displays the hexadecimal error code that indicates why the ER bit was set in the control data file (R6:). See page 11–27 for error code descriptions.

Example



If input slot 1, bit 10 is set, read 50 characters from the buffer and move to ST52:76.

When the rung goes from false-to-true, the Enable bit (EN) is set. When the instruction is placed in the ASCII queue, the Queue bit (EU) is set. The Running bit (RN) is set when the instruction is executing. The DN bit is set on completion of the instruction.

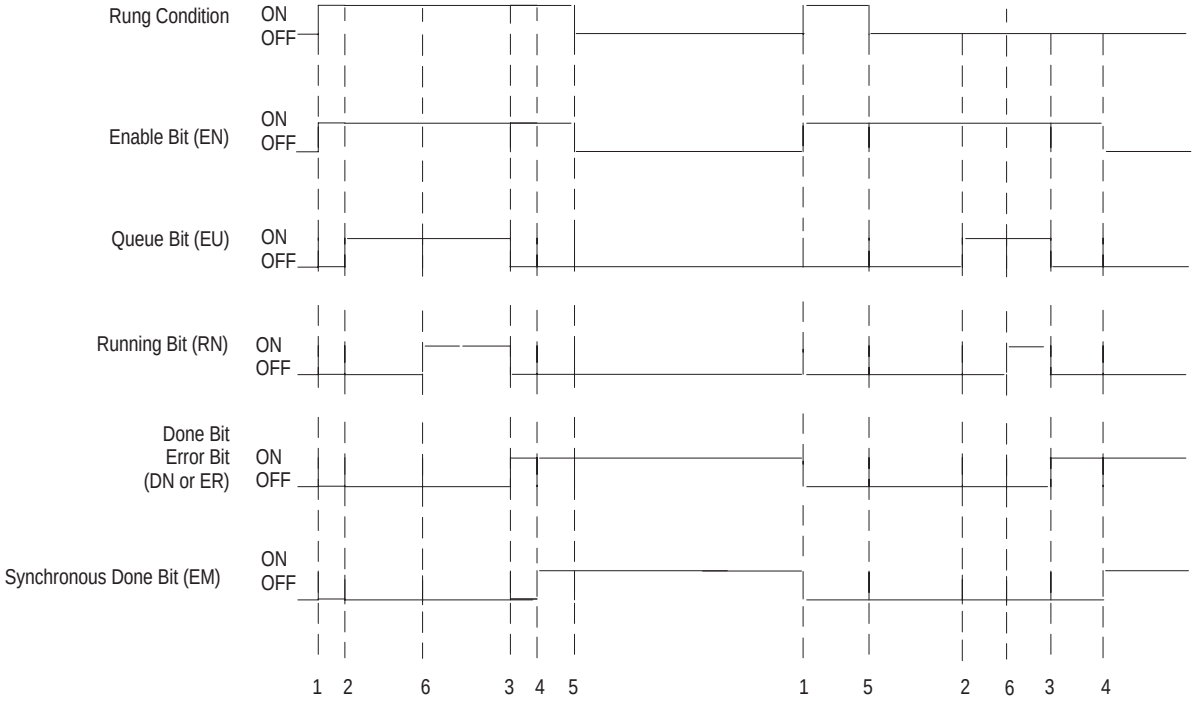
Once the requested number of characters are in the buffer, the characters are moved to the destination string. The number of characters moved is put in the POS field of the control block. The number in the Characters Read field is continuously updated and the Done bit (DN) is not set until all of the characters are read.

When the program scans the instruction and finds the Done bit (DN) set, the processor then sets the Synchronous Done bit (EM). The EM bit acts as a secondary done bit corresponding to the program scan.

The Error bit (ER) is set during the execution of the instruction if:

- the instruction is aborted – serial port is not in User mode
- the modem is disconnected (control line selection is other than “NO HANDSHAKING”)
- the instruction is aborted due to channel mode change
- the Unload bit (UL) is set. The instruction stops executing, but received characters are sent to the destination.
- an ACL to clear the receive buffer is executed, removing the ARD instruction from the ASCII queue

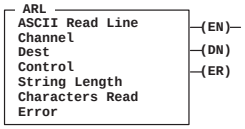
Timing Diagram for a Successful ARD, ARL, AWA, and AWT Instruction



1 - rung goes true
2 - instruction successfully queued
3 - instruction execution complete
4 - instruction scanned for the first time after execution is complete
5 - rung goes false
6 - either the instruction is not in the queue or it is being executed

ASCII Read Line (ARL)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

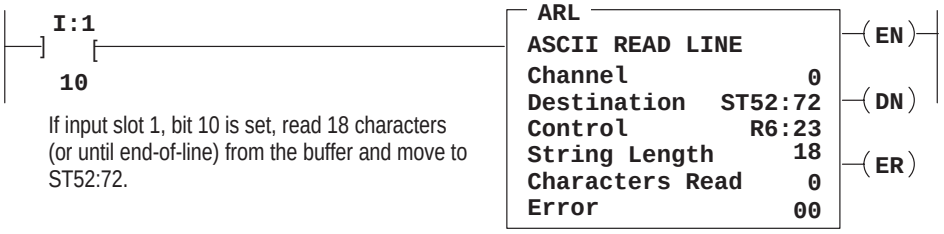
Use the ARL instruction to read characters from the buffer, up to and including the end-of-line (termination) characters, and store them in a string. The end-of-line characters are specified via the ASCII Configuration screen.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **Destination** is the string element where you want the characters stored.
- **Control** is the address of the control block used to store data for the ARL instruction.
- **String Length** (LEN) is the number of characters you want to read from the buffer. The maximum is 82 characters. If you specify a length larger than 82, only the first 82 characters are read and moved to the destination. (A 0 defaults to 82.) This is word 1 in the control block.
- **Characters Read** (POS) are the number of characters that the processor moved from the buffer to the string (0 to 82). This field is display only and resides in word 2 of the control block.
- **Error** displays the hexadecimal error code that indicates why the ER bit was set in the control data file (R6:). See page 11–27 for error code descriptions.

Example



If input slot 1, bit 10 is set, read 18 characters (or until end-of-line) from the buffer and move to ST52:72.

When the rung goes from false-to-true, the control element Enable (EN) bit is set. When the instruction is placed in the ASCII queue, the Queue bit (EU) is set. The Running bit (RN) is set when the instruction is executing**.** The DN bit is set on completion of the instruction.

Once the requested number of characters are in the buffer, all characters (including the end-of-line characters) are moved to the destination string. The number of characters moved is stored in the POS word of the control block. The number in the Characters Read field is continuously updated and the Done bit (DN) is not set until all of the characters have been read. Exception: If the processor finds termination characters before done reading, the Done bit (DN) is set and the number of characters found is stored in the POS word of the control block.

When the program scans the instruction and finds the Done bit (DN) set, the processor then sets the Synchronous bit (EM). The EM bit acts as a secondary done bit corresponding to the program scan.

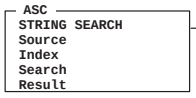
The Error bit (ER) is set during the execution of the instruction if:

- the instruction is aborted – serial port is not in User mode
- the modem is disconnected (control line selection is other than “NO HANDSHAKING”)
- the instruction is aborted due to channel mode change
- the Unload bit (UL) is set. The instruction stops executing, but received characters are sent to the destination.
- an ACL to clear the receive buffer is executed, removing the ARL instruction from the ASCII queue

Note *For information on the timing of this instruction, refer to the timing diagram on page 11–17.*

String Search (ASC)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

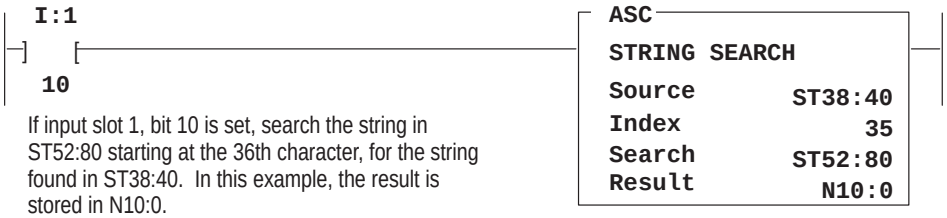
Use the ASC instruction to search an existing string for an occurrence of the source string.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source** is the string you want to find when examining the search string.
- **Index** is the starting position (from 1 to 82) of the portion of the string you want to find. (An index of 1 indicates the left-most character of the string.)
- **Search** is the string you want to examine.
- **Result** is an integer where the processor stores the position of the search string where the source string begins. If no match is found, result is set equal to zero.

Example



If input slot 1, bit 10 is set, search the string in ST52:80 starting at the 36th character, for the string found in ST38:40. In this example, the result is stored in N10:0.

The following conditions cause the processor to set the ASCII Error bit (S:5/15).

- invalid string length or string length of zero
- index value outside of range
- index value greater than the length of the source string

The destination is not changed in any of the above conditions.

ASCII String Compare (ASR)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

Input Instruction

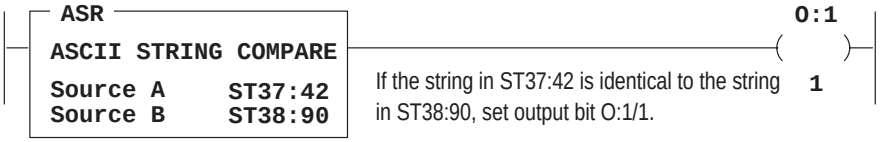
Use the ASR instruction to compare two ASCII strings. The system looks for a match in length and upper/lower case characters. If two strings are identical, the rung is true; if there are any differences, the rung is false.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Source A** is string one for comparison.
- **Source B** is string two for comparison.

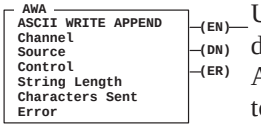
Example



An invalid string length causes the processor to set ASCII Error bit S:5/15, and the rung goes false.

ASCII Write with Append (AWA)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓



Output Instruction

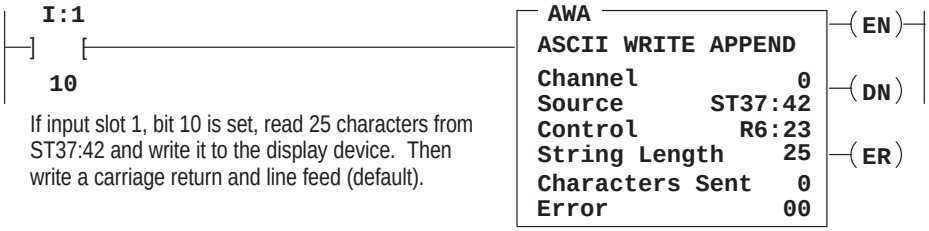
Use the AWA instruction to write characters from a source string to an external device. This instruction adds the two appended characters that you configure on the ASCII Configuration screen. The default is a carriage return and line feed appended to the end of the string. When using this instruction you can also perform in-line indirection. See page 11–24 for more information.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **Source** is the string element you want to write.
- **Control** is the area that stores the control register required to operate the instruction.
- **String Length** (.LEN) is the number of characters you want to write from the source string (0 to 82). If you enter a 0, the entire string will be written. This is word 1 in the control block.
- **Characters Sent** (.POS) are the number of characters that the processor sent to the display area (0 to 82). This field is continuously updated during the execution of the instruction. This value can be greater than the string length if appended characters or inserted values from in-line indirection are used. If the string length is greater than 82, the string written to the destination is truncated to 82 characters. This is word 2 in the control block.
- **Error** displays the hexadecimal error code that indicates why the ER bit was set in the control data file (R6:). See page 11–27 for error code descriptions.

Example



If input slot 1, bit 10 is set, read 25 characters from ST37:42 and write it to the display device. Then write a carriage return and line feed (default).

When the rung goes from false-to-true, the control element Enable (EN) bit is set. When the instruction is placed in the ASCII queue, the Queue bit (EU) is set. The Running bit (RN) is set when the instruction is executing**.** The DN bit is set on completion of the instruction.

The system sends 25 characters from the start of string ST37:42 to the display device and then sends user-configured append characters. The Done bit (DN) is set and a value of 27 is present in .POS word of the ASCII control block.

When the program scans the instruction and finds the Done bit (DN) set, the processor then sets the Synchronous Done bit (EM) to act as a secondary done bit corresponding to the program scan.

The Error bit (ER) is set during execution of the instruction if:

- the modem is disconnected (control line selection is other than “NO HANDSHAKING”)
- port is in System Mode and is configured for DH485
- the Unload bit (UL) is set. The instruction stops executing, but received characters are sent to the destination.
- an ACL to clear the send buffer is executed, removing the AWA instruction from the ASCII queue

Note *For information on the timing of this instruction, refer to the timing diagram on page 11–17.*

Using In-line Indirection

This allows you to insert integer and floating point values into ASCII strings. The Running bit (RN) must be set before the string value can be used.

The following conditions apply to performing in-line indirection:

- all valid integer (N) and floating point (F) files can be used. Valid ranges include 7, 8, and 9–255.
- file types are not case sensitive and can include either a colon (:) or semicolon (;)
- positive values and leading zeros are not printed. Negative values are printed with a leading minus sign.

Examples

For the following examples:
N7:0 = 250
N7:1 = –37
F8:0 = 2.015000
F8:1 = 0.873000

Valid in-line indirection:

- Input: Flow rate is currently [N7:0] GPH and contains [F8:0] PPM contaminants.
  Output: Flow rate is currently 250 GPH and contains 2.015000 PPM contaminants.
- Input: Current position is [N7:1] at a speed of [F8:1] RPM.
  Output: Current position is –37 at a speed of 0.873000 RPM.

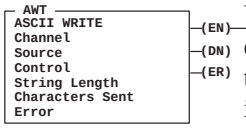
Invalid in-line indirection:

- Input: Current position is [N5:1] at a speed of [F8:1] RPM.
  Output: Current position is [N5:1] at a speed of 0.873000 RPM.

Note *Truncation occurs in the output string if the indirection causes the output to exceed 80 characters. The appended characters are always applied to the output.*

ASCII Write (AWT)

ML1000 Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓



Output Instruction

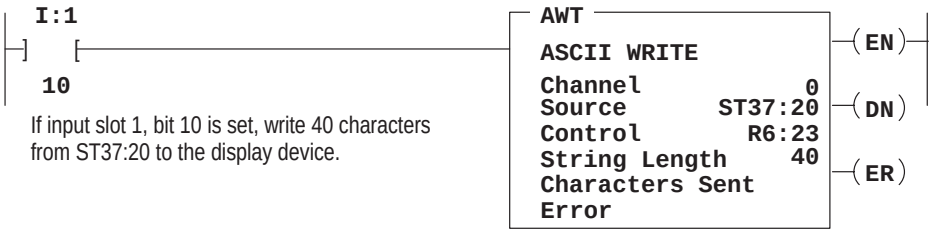
Use the AWT instruction to write characters from a source string to an external device. To repeat the instruction, the rung must go from false-to-true. When using this instruction you can also perform in-line indirection. See page 11–24 for more information.

Entering Parameters

Enter the following parameters when programming this instruction:

- **Channel** is the number of the RS-232 port (Channel 0).
- **Source** is the string element you want to write.
- **Control** is the area that stores the control register required to operate the instruction.
- **String Length** (LEN) is the number of characters you want to write from the source string (0 to 82). If you enter a 0, the entire string will be written.
- **Characters Sent** (POS) is the number of characters that the processor sent to the display area (0 to 82). Only after the entire string is sent is this field updated (no running total for each character is stored). This field is display only. This value can be greater than the string length if inserted values from in-line indirection are used. If the string length is greater than 82, the string written to the destination is truncated to 82 characters.
- **Error** displays the hexadecimal error code that indicates why the ER bit was set in the control data file (R6:). See page 11–27 for error code descriptions.

Example



If input slot 1, bit 10 is set, write 40 characters from ST37:20 to the display device.

When the rung goes from false-to-true, the control element Enable (EN) bit is set. When the instruction is placed in the ASCII queue, the Queue bit (EU) is set. The Running bit (RN) is set when the instruction is executing**.** The DN bit is set on completion of the instruction.

Forty characters from string ST37:40 are sent through channel 0. The Done bit (DN) is set and a value of 40 is present in the POS word of the ASCII control block.

When the program scans the instruction and finds the Done bit (DN) set, the processor then sets the Synchronous Done bit (EM) to act as a secondary done bit corresponding to the program scan.

The Error bit (ER) is set during execution of the instruction if:

- the modem is disconnected (control line selection is other than “NO HANDSHAKING”)
- port is in System Mode and is configured for DH485
- the Unload bit (UL) is set. The instruction stops executing, but received characters are sent to the destination.
- an ACL to clear the send buffer is executed, removing the AWT instruction from the ASCII queue

Note *For information on the timing of this instruction, refer to the timing diagram on page 11–17.*

ASCII Instruction Error Codes

The following error codes indicate why the Error bit (ER) is set in the control data file (R6:).

Error Code (HEX)	Conditions Resulting in the Setting of the ER Bit	Recommended Action
00	No error. The instruction completed successfully.	None required.
02	Operation cannot be completed because the modem went offline.	Check modem cabling to communication channel. If the channel is configured for modem handshaking, both the DCD (Data-Carrier-Detect) and DSR (Data-Set-Ready) lines to the channel must be active for the modem to be online.
03	Transmission cannot be completed because the Clear-to-Send signal was lost.	Check modem and modem cabling connections.
04	Cannot perform ASCII receives because the communication channel is configured for System Mode.	Reconfigure the communication channel for User Mode.
05	While attempting to perform ASCII transmission, System Mode (DF1) communication was detected.	Verify that the modem is online and communicating with required devices.
07	Cannot perform ASCII send or receive because channel configuration has been shut down via the channel configuration menu.	Reconfigure the channel configuration menu and retry operation.
08	Cannot perform ASCII write due to an ASCII transmission already in progress.	Resend the transmission.
09	ASCII communication requested is not supported by current channel configuration. (Channel 0 is configured for DH-485 while trying to initiate an ASCII transmission or modem handshake control.)	Configure channel 0 for DF1, Full-Duplex.
0A	The Unload bit (UL) was set, stopping instruction execution.	None required.
0B	The requested length for the string is either invalid, a negative number, greater than 82, or 0. Applies to ARD and ARL instructions.	Enter a valid string length and retry operation.
0C	The length of the source string is either invalid, a negative number, greater than 82, or 0. Applies to AWA and AWT instructions.	Enter a valid string length and retry operation.
0D	The requested length (.LEN) in the control block is a negative number or a value greater than the string size stored with the source string. Applies to AWA and AWT instructions.	Enter a valid length and retry operation.
0E	The ACL instruction was aborted.	None required.
0F	The channel configuration mode was changed.	None required.

ASCII Conversion Table

The table below lists the decimal, hexadecimal, octal, and ASCII conversions.

Column 1				Column 2				Column 3				Column 4			
DEC	HEX	OCT	ASC	DEC	HEX	OCT	ASC	DEC	HEX	OCT	ASC	DEC	HEX	OCT	ASC
00	00	000	NUL	32	20	040	SP	64	40	100	@	96	60	140	\
01	01	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
02	02	002	STX	34	22	042	”	66	42	102	B	98	62	142	b
03	03	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
04	04	004	EOT	36	24	044	$	68	44	104	D	100	64	144	d
05	05	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
06	06	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
07	07	007	BEL	39	27	047	’	71	47	107	G	103	67	147	g
08	08	010	BS	40	28	050	(	72	48	110	H	104	68	150	h
09	09	011	HT	41	29	051	)	73	49	111	I	105	69	151	i
10	0A	012	LF	42	2A	052	*	74	4A	112	J	106	6A	152	j
11	0B	013	VT	43	2B	053	+	75	4B	113	K	107	6B	153	k
12	0C	014	FF	44	2C	054	,	76	4C	114	L	108	6C	154	l
13	0D	015	CR	45	2D	055	–	77	4D	115	M	109	6D	155	m
14	0E	016	SO	46	2E	056	.	78	4E	116	N	110	6E	156	n
15	0F	017	SI	47	2F	057	/	79	4F	117	O	111	6F	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1A	032	SUB	58	3A	072	:	90	5A	132	Z	122	7A	172	z
27	1B	033	ESC	59	3B	073	;	91	5B	133	[	123	7B	173	{
28	1C	034	FS	60	3C	074	<	92	5C	134	\	124	7C	174	.
29	1D	035	GS	61	3D	075	=	93	5D	135	]	125	7D	175	}
30	1E	036	RS	62	3E	076	>	94	5E	135	^	126	7E	176	~
31	1F	037	US	63	3F	077	?	95	5F	137	_	127	7F	177	DEL

12 Understanding Interrupt Routines

This chapter contains general information about interrupt routines and explains how they function in your logic program. Each interrupt routine includes:

- an overview
- programming procedure
- operational description
- associated bit description

In addition, each interrupt routine contains an application example that shows the interrupt routine in use.

Interrupt Routines

Instruction		Purpose	Page
Mnemonic	Name		
	User Fault Routine	Provides the option of preventing a processor shut-down.	12–2
STI	Selectable Timed Interrupt	Allows you to interrupt the scan of the main program file automatically, on a periodic basis, to scan a specified subroutine file.	12–7
DII	Discrete Input Interrupt	Allows the processor to execute a subroutine when the input bit pattern of a discrete I/O card matches a compare value that you programmed.	12–17
ISR	I/O Interrupt	Allows a specialty I/O module to interrupt the normal processor operating cycle in order to scan a specified subroutine file.	12–27

User Fault Routine Overview

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓

The user fault routine gives you the option of preventing a processor shutdown when a specific user fault occurs. The file is executed when any recoverable or non-recoverable user fault occurs. The file is not executed for non-user faults.

You do this by programming a ladder subroutine, then specifying that subroutine as the fault routine in word S:29 in the status file. You can handle a number of user faults in this way, as the example on page 12–4 shows.

Faults are classified as recoverable and non-recoverable user faults, and non-user faults. A complete list of faults appear in appendix A and B for the MicroLogix 1000 controllers and the SLC processors respectively.

Non–User Fault	Non–Recoverable User Fault	Recoverable User Fault
The Fault Routine does not execute.	The Fault Routine executes for 1 pass. **Note:** You may initiate a MSG instruction to another node to identify the fault condition of the processor.	The Fault Routine may clear the fault by clearing bit S:1/13.

Status File Data Saved

Data in the following words is saved on entry to the user fault subroutine and re-written upon exiting the subroutine.

- S:0 Arithmetic flags
- S:13 and S:14 Math register
- S:24 Index register

Creating a User Fault Subroutine

To use the user fault subroutine:

1. Create a subroutine file:
   - SLC processor valid range is 3–255
   - Micro Logix 1000 designates File 3
2. Enter the file number in word S:29 of the status file.

Note *No action is required for MicroLogix 1000 users. S:29 is reserved.*

SLC Processor Operation

The occurrence of recoverable or non-recoverable user faults causes the processor to read S:29 and execute the subroutine number contained in S:29. If the fault is recoverable, the routine can be used to correct the problem and clear the fault bit S:1/13. The processor then continues in the REM Run mode.

The routine does not execute for non-user faults.

Words S:20 and S:21 can be examined in your fault routine to pinpoint the file and rung number where the fault occurred. If the fault occurred outside of the ladder scan, this value will contain the rung number where the TND, END, or REF instruction is located. Use words S:20 and S:21 with your powerup protection fault routine to determine the exact point that the previous power down occurred. Refer to appendix B for more information about the Startup Protection Fault bit, S:1/9.

Note *For SLC 5/02 processors, you must save your program with test single step selected in order for S:20 and S:21 to be activated.*

For SLC 5/03 and higher processors, if your program contains four message instructions with the Continuous Operation (CO) bit set, the fault routine's message instruction is not executed.

MicroLogix Controller Operation

The occurrence of recoverable or non-recoverable user faults causes file 3 to be executed. If the fault is recoverable, the routine can be used to correct the problem and clear the fault bit S:1/13. The processor then continues in the REM Run mode.

The routine does not execute for non-user faults.

User Interrupt Routine Application Example

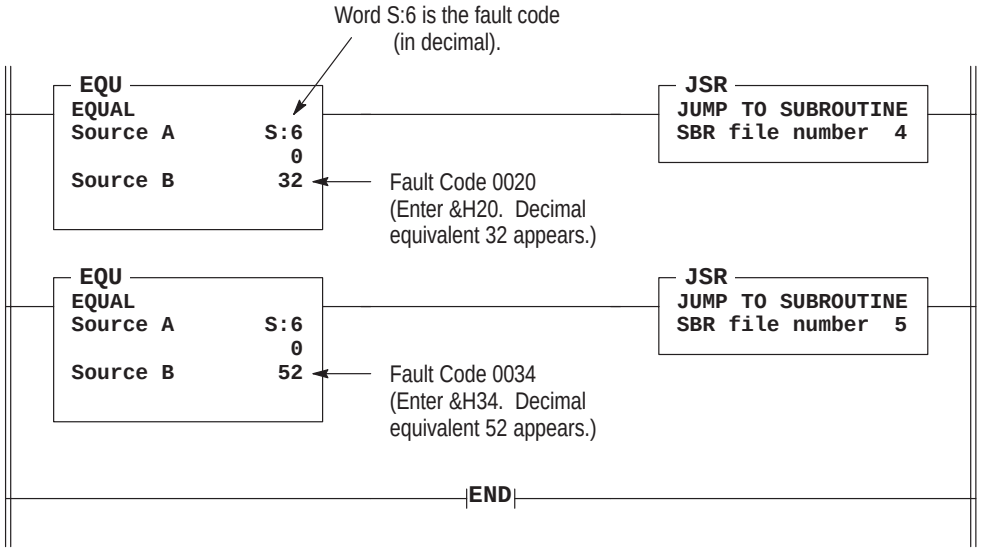
Suppose you have a program in which you want to control major errors 0020 **(MINOR ERROR AT END OF SCAN)** and 0034 **(NEGATIVE VALUE IN TIMER PRE OR ACC)** under the following conditions:

- Prevent a processor shutdown if the overflow trap bit S:5/0 is set. Permit a processor shutdown when S:5/0 is set more than five times.
- Prevent a processor shutdown if the accumulator value of timer T4:0 becomes negative. Reset the negative accumulator value to zero. Energize an output to indicate that the accumulator has gone negative one or more times.
- Allow a processor shutdown for all other user faults.

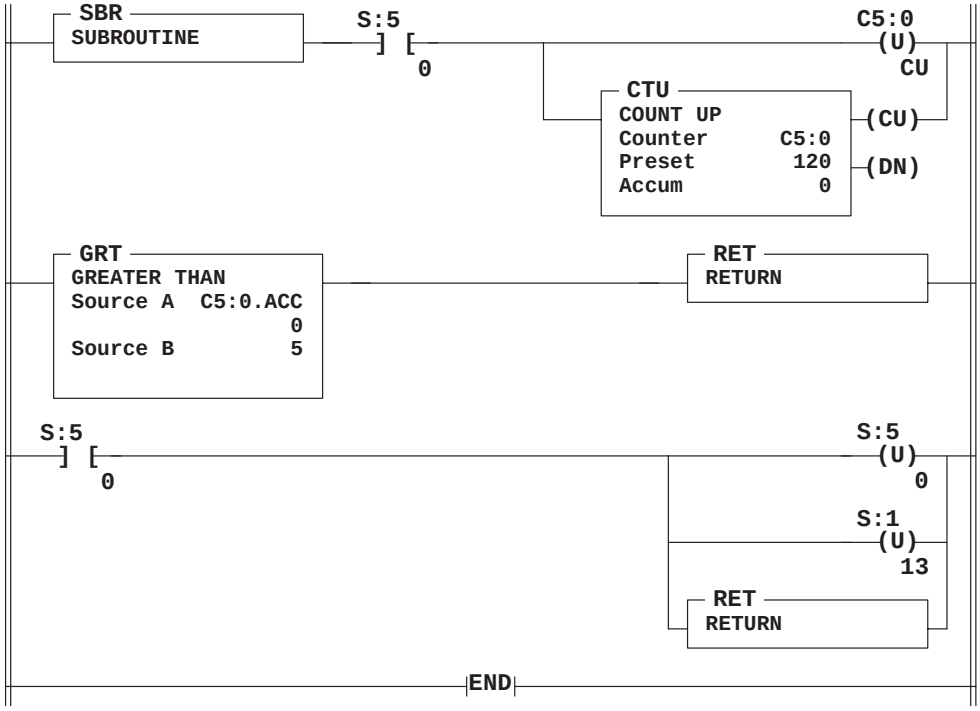
A possible method of accomplishing this is shown in the following examples. The user fault routine is designated as file 3.

When a recoverable or non-recoverable user error occurs, the processor scans subroutine file 3. The processor jumps to file 4 if the error code is 0020 and it jumps to file 5 if the error code is 0034. For all other recoverable and non-recoverable errors, the processor exits the fault routine and halts operation in the fault mode.

Fault Routine – Subroutine File 3



Subroutine File 4 – Executed for Error 0020

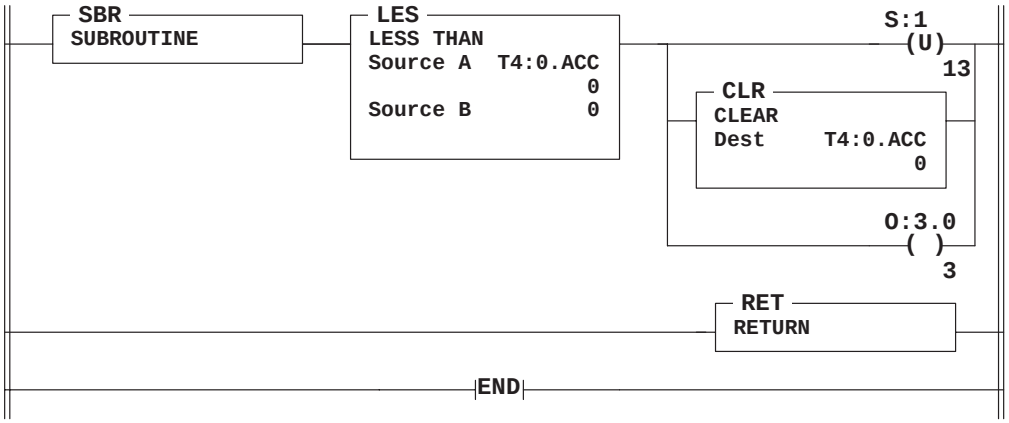


If the overflow trap bit, S:5/0 is set, counter C5:0 increments.

If the count of C5:0 is 5 or less, the overflow trap, S:5/0 is cleared, the major error halted bit S:1/13 is cleared, and the processor remains in the REM Run mode. If the count is greater than 5, the processor sets S:5/0 and S:1/13 and enters the Fault mode.

This subroutine file is also executed if the control register error bit S:5/2 is set. In this case, the processor is placed in the Fault mode.

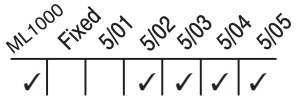
Subroutine File 5 – Executed for Error 0034



If the accumulator value of timer T4:0 is negative, the major error halted bit, S:1/13 is unlatched, preventing the processor from entering the Fault mode. At the same time, the accumulator value T4:0 ACC is cleared to zero and output O:3.0/3 is energized. Fault code 0034 is displayed in the status file.

If the preset of timer T4:0 is negative, S:1/13 remains set and the processor enters the Fault mode (O:3.0/3 will be reset if previously set). Also, if either the preset or accumulator value of any other timer in the program is negative, S:1/13 is set and the processor enters the Fault mode. If previously set, O:3.0/3 is reset.

Selectable Timed Interrupt Overview



This function allows you to interrupt the scan of the processor automatically, on a periodic basis, to scan a specified subroutine file. Afterward, the processor resumes executing from the point where it was interrupted.

This section describes:

- STI programming procedure
- STI operation and parameters
- STD and STE instructions
- STS instruction

Basic Programming Procedure for the STI Function

To use the STI function in your application file:

1. Create a subroutine file and enter the desired ladder rungs. This is your STI subroutine file.
   - SLC processors valid range is 3–255
   - MicroLogix 1000 controllers designate File 5
2. SLC processors – Enter the STI subroutine file number in word S:31 of the status file. Refer to page B–48 in this manual for more information. A file number of zero disables the STI function.
3. Enter the setpoint (the time between successive interrupts) in word S:30 of the status file. Refer to page A–20 for MicroLogix 1000 controllers or B–47 for SLC processors for more information.
   - For SLC 5/02 and MicroLogix 1000 controllers, the range is 10–2550 ms (entered in 10 ms increments). A setpoint of zero disables the STI function.
   - For SLC 5/03 and higher processors, the range is from 1–32,767 ms (entered in 1 ms increments). A setpoint of zero disables the STI function. Refer to appendix B in this manual for more information about the STI Resolution bit S:2/10.

Note *The setpoint value must be a longer time than the execution time of the STI subroutine file, or a minor error bit is set. For all processors, the STI Pending bit and STI Overrun bit will be set. Additionally, for the SLC 5/03 and higher processors, and MicrLogix 1000 controllers, the STI Last bit may be set.*

Operation

After you restore your program and enter the REM Run mode, the STI begins operation as follows:

1. The STI timer begins timing.
2. When the STI interval expires, the STI timer is reset, the processor scan is interrupted and the STI subroutine file is scanned.
3. If while executing the STI subroutine, another STI interrupt occurs, the STI pending bit is set.
4. If while an STI is pending, the STI timer expires, the STI lost bit is set. (For SLC 5/02 processors, the Overrun bit is set.)
5. When the STI subroutine scan is completed, scanning of the main program file resumes at the point where it left off, unless an STI is pending. In this case, the subroutine is immediately scanned again.
6. The cycle repeats.

For identification of your STI subroutine, include an INT instruction as the first instruction on the first rung of the file.

STI Subroutine Content

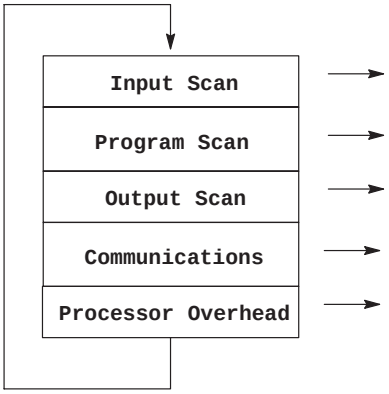
The STI subroutine contains the rungs of your application logic. You can program any instruction inside the STI subroutine except a TND, REF, or SVC instruction. IIM or IOM instructions are needed in an STI subroutine if your application requires immediate update of input or output points. End the STI subroutine with an RET instruction.

JSR stack depth is limited to 3. You may call other subroutines to a level 3 deep from an STI subroutine.

Interrupt Latency and Interrupt Occurrences

Interrupt latency is the interval between the STI timeout and the start of the interrupt subroutine. STI interrupts can occur at any point in your program, but not necessarily at the same point on successive interrupts. The tables below show the interaction between an interrupt and the processor operating cycle.

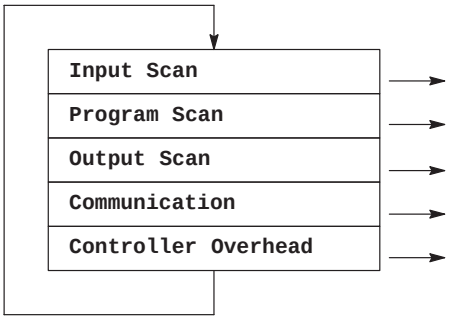
SLC Processors



SLC 5/02 STI	SLC 5/03 and Higher STI with Bit S:33/8 set	SLC 5/03 and Higher STI with Bit S:33/8 cleared
Between slot updates	Between word updates	Between slot updates
Between instruction updates	Between word updates	Between rung updates
Between slot updates	Between word updates	Between slot updates
Between communication packets	Between word packet updates	Between communication packets
At start and end	Between word updates	Between word updates

Events in the Processor Operating Cycle

MicroLogix Controller



STI
Between instruction updates
Between communication packets
At start and end

Events in the processor operating cycle

Note that STI execution time adds directly to the overall scan time. During the latency period, the processor is performing operations that cannot be disturbed by the STI interrupt function.

Latency periods are:

- SLC 5/02 processors and MicroLogix 1000 controllers interrupts are serviced within 2.4 ms maximum.
- SLC 5/03 and higher processors – If an interrupt occurs while the processor is performing a multi–word slot update and your interrupt subroutine accesses that same slot, the multi–word transfer finishes to completion prior to performing the interrupt subroutine slot access. The Interrupt Latency Control bit (S:33/8) functions as follows:
  - When the bit is set (1), interrupts are serviced within the interrupt latency time. Refer to appendix D for more information on how to calculate the interrupt latency.
  - When the bit is clear (0), INTs are serviced per rung, slot, and packet execution time.

  The default state is cleared (0). To determine the interrupt latency with S:33/8 clear, you must calculate the execution time of each and every rung in your program. Use the longest calculated execution time plus 500µs as your maximum interrupt latency.

Interrupt Priorities

Interrupt priorities for the processors are:

MicroLogix 1000 Controller	SLC 5/02 Processor	SLC 5/03 and Higher Processors
1. User Fault Routine	1. User Fault Routine	1. User Fault Routine
2. High–Speed Counter	2. Selectable Timed Interrupt Subroutine	2. Discrete Input Interrupt (DII)
3. Selectable Timed Interrupt Subroutine	3. Interrupt Subroutine (ISR)	3. Selectable Timed Interrupt Subroutine
		4. Interrupt Subroutine (ISR)

An executing interrupt can only be interrupted by an interrupt having higher priority.

Status File Data Saved

Data in the following words is saved on entry to the STI subroutine and re-written upon exiting the STI subroutine.

- S:0 Arithmetic flags
- S:13 and S:14 Math register
- S:24 Index register

STI Parameters

The following parameters are associated with the STI function. These parameters have status file addresses that are described here and also in appendix A and appendix B of this manual.

- **STI file number (Word S:31)** – This can be any number from 3–255. A value of zero disables the STI function. An invalid number generates fault 0023. *This word does not apply to MicroLogix 1000 controllers.*
- **Setpoint (Word S:30)** – This is the time between the starting point of successive scans of the STI file. It can be any value from 10 to 2550 milliseconds. You enter a value of 1 to 255, which results in a 10–2550 ms setpoint. A value of zero disables the STI function. An invalid time generates fault 0024.

  If the STI is initiated while in the REM Run mode by loading the status registers, the interrupt will start timing from the end of the program scan in which the status registers were loaded.

  SLC 5/03 and higher processors – If S:2/10 is set, time is in 1 ms increments. If this bit is clear, time is in 10 ms increments.

- **STI Pending Bit (S:2/0)** – This bit is set when the STI timer has timed out and the STI routine is waiting to be executed. This bit is reset upon starting the STI routine, execution of a true STS instruction, powerup, or exit from the REM Run or Test mode.

  SLC 5/02 specific – The STI pending bit is not set if the STI timer expires while executing the fault routine.

  SLC 5/03 and higher processors – This bit is set if the STI timer expires while executing the DII subroutine or fault routine.

- **STI Enable Bit (S:2/1)** – The default value is 1 (set). When a file number between 3 and 255 is present in word S:31 and a setpoint value between 1 and 255 is present in word S:30, a set enable bit allows scanning of the STI file. If the bit is reset by an STD instruction, scanning of the STI file no longer occurs. If the bit is set by an STE or STS instruction, scanning is again allowed. The enable bit only enables/disables the scanning of the STI subroutine. It does not affect the STI timer. The STS instruction affects both the enable bit and the STI timer. The default state is enabled. If this bit is set or reset using the STE, STD, or STS instruction, enable/disable takes effect immediately. If this bit is set in the user program using an instruction other than STE, STD, or STS, it takes effect at the next end of scan.

  MicroLogix 1000 controller – This bit is set or reset using an STS, STE, or STD instruction. If set, it allows execution of the STI if the STI setpoint S:30 is non–zero. If clear, when an interrupt occurs, the STI subroutine does not execute and the STI Pending bit is set.

  SLC 5/02 specific – If this bit is set or reset by the user program or communications, it does not take effect until the next end of scan.

  SLC 5/03 and higher processors – If this bit is set or reset by the user program or communications, it takes effect upon the STI timer expiration or next end of scan (whichever occurs first).

- **STI Executing Bit (S:2/2)** – This bit is set when the STI file is being scanned and cleared when the scan is completed. The bit is also cleared on powerup and entry into the REM Run mode.
- **STI Resolution Selection Bit (S:2/10)** – This bit is clear by default. When clear, this bit selects a 10 ms increment for the STI Setpoint (S:30) value. When set, this bit selects a 1 ms increment for the STI Setpoint (S:30) value. To program this feature, use the data monitor function to set/clear this bit, or address this bit with your ladder program.

  This bit is user configurable and takes effect on a REM PROG to REM RUN mode transition.

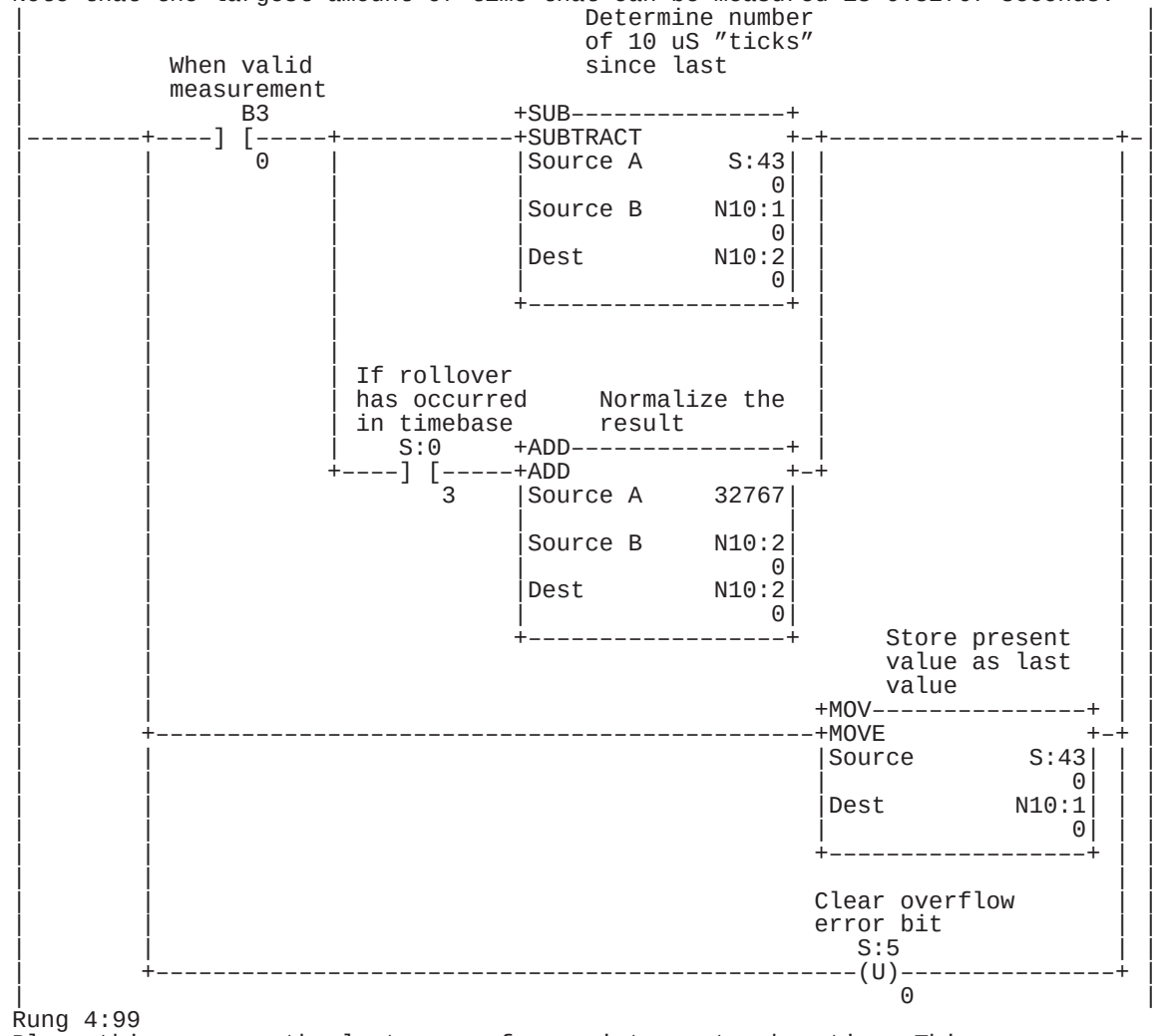
- **Overrun Bit (S:5/10)** – This minor error bit is set whenever the STI timer expires while the STI routine is executing or disabled while the pending bit is set. When this occurs, the STI timer continues to operate at the rate present in word S:30. If the overrun bit becomes set, take the corrective action your application dictates, then clear the bit.

- **STI Lost Bit (Word S:36/9)** – This bit is set anytime an STI interrupt occurs while the STI Pending bit is also set. When set, you are notified that a STI interrupt has been lost. For example, the interrupt is lost because a previous interrupt was already pending and waiting execution. Examine this bit in your user program and take appropriate action if your application cannot tolerate this condition. Then clear this bit with your user program in order to prepare for the next possible occurrence of this error.

Use the following rungs to initialize and measure the amount of time between two consecutive STI subroutine executions. The 10 µs timer is also available in the DII interrupt and I/O interrupt. This application example can also be used for the Event I/O interrupt or the DII interrupt by replacing S:43 with either S:44 or S:45 respectively.

```
Program Listing           Processor File: FREESTI.ACH                  Rung 2:0

Rung 2:0
Place this rung as the first rung of your main ladder program (file 2 rung 0).
This rung ensures that the interrupt measurement is initialized each time
the run mode is entered.
| 1st                                                    Indicate               |
| Pass                                                   measurment             |
|                                                        initialization         |
|                                                                               |
|    S:1                                                      B3                |
|----] [-----------------------------------------------+----(U)---------------+-|
|      15                                             |       0               | |
|                                                     |      Clear            | |
|                                                     |      10 uS            | |
|                                                     |      "tick"           | |
|                                                     |      Register         | |
|                                                     | +MOV---------------+  | |
|                                                     +-+MOVE               +-+ |
|                                                       |Source           0|   |
|                                                       |                  |   |
|                                                       |Dest         N10:2|   |
|                                                       |                 0|   |
|                                                       +------------------+   |


ladder program continued on next page
```

Rung 4:0
This rung measures the time between consecutive interrupt subroutine executions.
Integer N10:2 contains the number of 10 microsecond ”ticks” that have occured.
Note that the largest amount of time that can be measured is 0.32767 seconds.



Rung 4:99
Place this rung as the last rung of your interrupt subroutine. This way your interrupt subroutine will know when the value in N10:2 is valid.

Note *The math overflow selection bit (S:2/14) must be set prior to entering RUN mode.*

STD and STE Instructions

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓

The STD and STE instructions are used to create zones in which STI interrupts *cannot* occur.

Selectable Timed Disable – STD

When true, this instruction resets the STI enable bit and prevents the STI subroutine from executing. When the rung goes false, the STI enable bit remains reset until a true STS or STE instruction is executed. The STI timer continues to operate while the enable bit is reset.

Selectable Timed Enable – STE

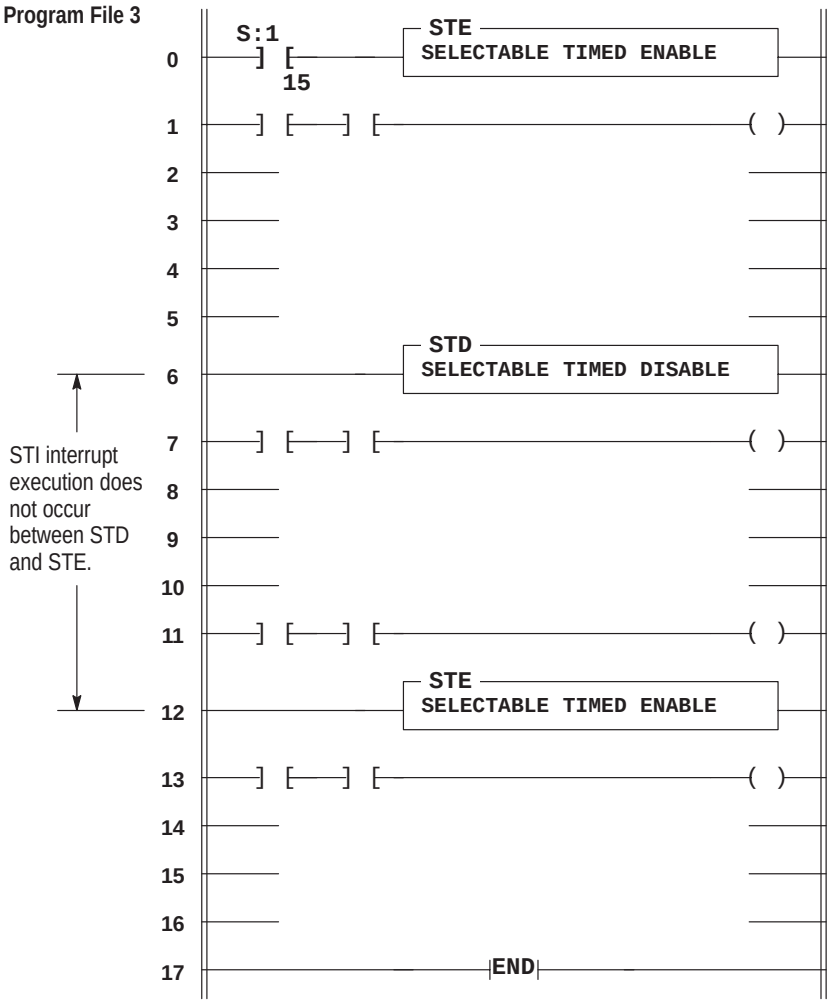
This instruction, upon a false-true transition of the rung, sets the STI enable bit and allows execution of the STI subroutine. When the rung goes false, the STI enable bit remains set until a true STD instruction is executed. This instruction has no effect on the operation of the STI timer or setpoint. When the enable bit is set, the first execution of the STI subroutine can occur at any fraction of the timing cycle up to a full timing cycle later.

STD/STE Zone Example

In the program that follows, the STI function is in effect. The STD and STE instructions in rungs 6 and 12 are included in the ladder program to avoid having STI subroutine execution at any point in rungs 7 through 11.

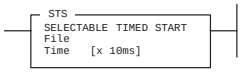
The STD instruction (rung 6) resets the STI enable bit and the STE instruction (rung 12) sets the enable bit again. The STI timer increments and may time out in the STD zone, setting the pending bit S:2/0 and overrun bit S:5/10.

The first pass bit S:1/15 and the STE instruction in rung 0 are included to insure that the STI function is initialized following a power cycle. You should include this rung any time your program contains an STD/STE zone or an STD instruction.


Program File 3
S:1
15
STE
SELECTABLE TIMED ENABLE
STD
SELECTABLE TIMED DISABLE
STE
SELECTABLE TIMED ENABLE
END
STI interrupt execution does not occur between STD and STE.
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17

Selectable Timed Start (STS)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓



Use the STS instruction to condition the start of the STI timer upon entering the REM Run mode – rather than starting automatically. You can also use it to set up or change the file number or setpoint/frequency of the STI routine that is executed when the STI timer expires.

This instruction is not required to configure a basic STI interrupt application.

The STS instruction requires you to enter two parameters, the STI file number and the STI setpoint. Upon a true execution of the rung, this instruction enters the file number and setpoint in the status file (S:31, S:30), overwriting the existing data. At the same time, the STI timer is reset and begins timing; at timeout, the STI subroutine execution occurs. When the rung goes false, the STI function remains enabled at the setpoint and file number you've entered in the STS instruction.

Note *SLC 5/03 and higher processors –The STS instruction uses the setting of the STI resolution bit S:2/10 to determine the timebase to be used upon STS instruction execution.*

Discrete Input Interrupt Overview

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

Use the Discrete Input Interrupt (DII) for high–speed processing applications or any application that needs to respond to an event quickly. This instruction allows the processor to execute a ladder subroutine when the input bit pattern of a discrete I/O card matches a compare value that you programmed.

The status file contains six bit values and six word values used to program and monitor the DII function. The DII does not require ladder logic instructions for configuration. You program the DII to examine the input bit pattern of any single I/O slot, which contains any discrete input card (such as IG16, IV16, IB8, IB32). When the input bit pattern matches the compare value, the accumulator is incremented. The DII accumulator counts to the preset value and, once the interrupt is generated, it immediately wraps around and begins counting again at zero.

While scanning the DII subroutine, you can reconfigure the DII to look for an entirely different event. This facilitates DII sequencing. The DII can be programmed to compare each input point to either a high (1) or low (0) state. The accumulator is incremented on the input transition that causes the input points to match the compare value.

IIM or IOM instructions are needed in the DII subroutine if your application requires immediate update of input or output points. End the DII subroutine with an RET instruction.

Basic Programming Procedure for the DII Function

To use the DII function with your main program file, do the following:

1. Create a subroutine file (range is from 3 to 255) and enter the desired ladder rungs. This is your DII subroutine file.
2. Enter the Input Slot number (word S:47).
3. Enter the Bit Mask (word S:48).
4. Enter the Compare Value (word S:49).
5. Enter the Preset Value (word S:50).
6. Enter the DII subroutine file number in word S:46 of the status file. (See page B–58.) A zero value disables the DII function.

Note

PLC users – *The main difference between the DII and the PLC 5/40 PII is that the DII requires all stated transitions to occur prior to generating a count, while the PII requires that only one of the stated transitions occur. Also, the PLC term "count" is referred to as "preset" in the DII.*

Example

The DII can be programmed to count items on a high–speed conveyer. Each time 100 items pass a photo-switch, the DII subroutine is executed. The DII subroutine then uses Immediate I/O instructions to package the products.

Operation

After you restore your program and enter the REM Run mode, the DII begins operation as follows:

Counter Mode

This mode is active when the preset value (S:50) contains a value greater than 1.

1. The DII reads the first byte of input data of a selected discrete input card at least once every 100µs.① Note that this "polling" of the input data has no effect on processor scan time.
2. When the input data matches the programmed masked value, the accumulator is incremented by one. The next count occurs when input data transitions to non–matched and then back to matched.
3. When the accumulator reaches or exceeds the preset value, between 1 and 32,767, the interrupt is generated and the accumulator is reset to zero.
4. The DII subroutine is executed.
5. The cycle repeats.

Event Mode

This mode is active when the preset value (S:50) contains a 0 or 1.

1. The DII reads the first byte of input data of a selected discrete input card at least once every 100µs.① Note that this "polling" of the input data has no effect on processor scan time.
2. When the input data matches the programmed masked value, the interrupt is generated.
3. The DII subroutine is executed.②
4. The cycle repeats.①

① You must add interrupt latency time to the final transition or count that causes the interrupt subroutine to execute. See page D–3 for calculating DII latency.

② The DII continues to compare the input data to the programmed masked value while executing the DII subroutine.

DII Subroutine Content

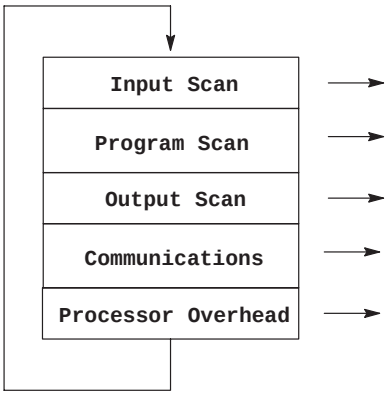
For identification of your DII subroutine, use the INT instruction as the first instruction in your first rung.

The DII subroutine contains the rungs of your application logic. You can program any instruction inside the DII subroutine except a TND, REF, or SVC instruction. IIM or IOM instructions are needed in a DII subroutine if your application requires immediate update of input or output points. End the DII subroutine with an RET instruction.

JSR stack depth is limited to 3. You may call other subroutines to a level 3 deep from an DII subroutine.

Interrupt Latency and Interrupt Occurrences

Interrupt latency is the interval between DII detection and the start of the interrupt subroutine. DII interrupts can occur at any point in your program, but not necessarily at the same point on successive interrupts. Interrupts can occur between instructions in your program, inside the I/O scan (between slots), or between the servicing of communications packets. The table below shows the interaction between an interrupt and the processor operating cycle.



Events in the Processor Operating Cycle

	DII	DII with Bit S:33/8 set	DII with Bit S:33/8 cleared
Input Scan →	Between slot updates	Between word updates	Between slot updates
Program Scan →	Between instruction updates	Between word updates	Between rung updates
Output Scan →	Between slot updates	Between word updates	Between slot updates
Communications →	Between communication packets	Between word packet updates	Between communication packets
Processor Overhead →	At start and end	Between word updates	Between word updates

If an interrupt occurs while the SLC 5/03 (or higher) processor is performing a multi-word slot update and your interrupt subroutine accesses that same slot, the multi-word transfer completes prior to performing the interrupt subroutine slot access.

Note that DII execution time adds directly to the overall scan time. During the latency period, the processor is performing operations that cannot be disturbed by the DII interrupt function. The Interrupt Latency Control Bit (S:33/8) functions as follows:

- When the bit is set (1) interrupts are serviced within the minumum time possible. The time will vary depending upon which processor and communication protocol you are using. Refer to appendix D for information on how to calculate interrupt latency when S:33/8 = 1.
- The default state is cleared (0). When S:33/8 is clear (0), user interrupts occur between rungs and I/O slot updates. To determine the interrupt latency with S:33/8 clear, you must calculate the execution time of each and every rung of your program, then the add the execution time of the longest rung to the latency time.

Interrupt Priorities

Interrupt priorities for the SLC 5/03 and higher processors are:

1. User fault routine
2. Discrete Input Interrupt (DII)
3. STI Subroutine
4. I/O Interrupt Subroutine

An executing interrupt subroutine can only be interrupted by the fault routine.

Status File Data Saved

Data in the following words is saved on entry to the DII subroutine and re-written upon exiting the DII subroutine.

- S:0 Arithmetic flags
- S:13 and S:14 Math register
- S:24 Index register

Reconfigurability

You can reconfigure the DII entirely or in part, depending on the particular parameter(s) you choose. You can reconfigure some of the parameters simply by writing the new value over the old value. Other values require you to set the reconfiguration bit in addition to writing the new value. The DII is non–retentive and always reconfigures itself upon entry into the REM Run mode. Refer to the next section “DII Parameters” for details on reconfiguring each parameter.

Example

The DII can be programmed to count items on a high–speed conveyer. Each time 100 items pass a photo-switch, the DII subroutine is executed. The DII subroutine then uses Immediate I/O instructions to package the products.

If you want to vary the number of items that are packaged together, simply change the number in the DII preset parameter using a MOV instruction.

DII Parameters

The following parameters are associated with the DII function. These parameters have status file addresses that are described here and also in appendix B.

- **DII Pending Bit (S:2/11)** – When set, this bit indicates that the DII accumulator (S:52) equals the DII preset (S:50) and the ladder file number specified by the DII file number (S:46) is waiting to be executed. It is cleared when the DII file number (S:46) begins executing, or on exit from the REM Run or REM Test mode.
- **DII Enable Bit (S:2/12)** – To program this feature, use the data monitor function to set/clear this bit, or address this bit with your ladder program. This bit is set in its default condition. If set, it allows execution of the DII subroutine if the DII file (S:46) is non–zero. If clear, when the interrupt occurs, the DII subroutine will not execute and the DII Pending bit is be set. The DII function continues to run anytime the DII file (S:46) is non–zero. If the pending bit is set, the enable bit is examined at the next end of scan.
- **DII Executing Bit (S:2/13)** – When set, this bit indicates that the DII interrupt has occurred and the DII subroutine is currently being executed. This bit is cleared on completion of the DII routine, powerup, or REM Run mode entry.
- **DII Overflow Bit (S:5/12)** – This bit is set whenever the DII interrupt occurs while still executing the DII subroutine or whenever the DII interrupt occurs while pending or disabled.
- **Reconfigure Bit (S:33/10)** – When this bit is set (1), it indicates that at the next end of scan (END, TND, or REF), fault routine exit, STI ISR exit, Event ISR exit, or next DII ISR exit the:
  - DII accumulator is cleared,
  - values at status words S:47 to S:50 are applied,
  - the pending bit is cleared, and
  - the DII Reconfigure bit is cleared.
- **DII Lost Bit (S:36/8)** – This bit is set if a DII interrupt occurs while the DII Pending bit is set.
- **File Number (Word S:46)** – You enter a program file number (3 to 255) to be used as the discrete input interrupt subroutine. Write a 0 value to disable the function. This value is applied upon detection of a DII Reconfigure bit, each DII ISR exit, and each end of scan (END, TND, or REF). A zero disables operation.
- **Slot Number (Word S:47)** – You enter the slot number (1 to 30) to be used as the discrete input interrupt subroutine. A zero value disables the function. This value is applied on detection of the DII Reconfigure bit, or on entry into the REM Run mode.

- **Bit Mask (Word S:48)** – You enter the bit-mapped value that corresponds to the bits you wish to monitor on the discrete I/O module. Only bits 0 to 7 are used in the DII function. Setting a bit indicates that you wish to include the bit in the comparison of the discrete I/O card's bit pattern to the DII compare value (S:49). This value is applied on detection of the DII Reconfigure bit, each DII ISR exit, and at each end of scan (END, TND, or REF).
- **Compare Value (Word S:49)** – You enter a bit-mapped value that corresponds to the bit pattern that must occur on the discrete I/O card for a count or interrupt to occur. Only bit 0 to 7 are used in the DII function. The bit must be set (1) or cleared (0) in order to satisfy the compare condition for that bit. An interrupt or count is generated upon the last bit transition of the compare value. This value is applied on detection of DII Reconfigure bit, each DII ISR exit, and at each end of scan (END, TND, or REF).

  To provide protection from inadvertent data monitor alteration of your selection, program an unconditional MOV instruction containing the compare value of the DII to S:49.

- **Preset** (**Word S:50)** – When this value is equal to 0 or 1, an interrupt is generated each time the comparison specified in words S:48 and S:49 is satisfied. When this value is between 2 and 32767, a count occurs each time the bit comparison is satisfied. An interrupt is generated when the accumulator value reaches 1 or exceeds the preset value. This value is applied on detection of DII Reconfigure bit, each DII ISR exit, and at each end of scan (END, TND, or REF).

  To provide protection from inadvertent data monitor alteration of your selection, program an unconditional MOV instruction containing the preset value of the DII to S:50.

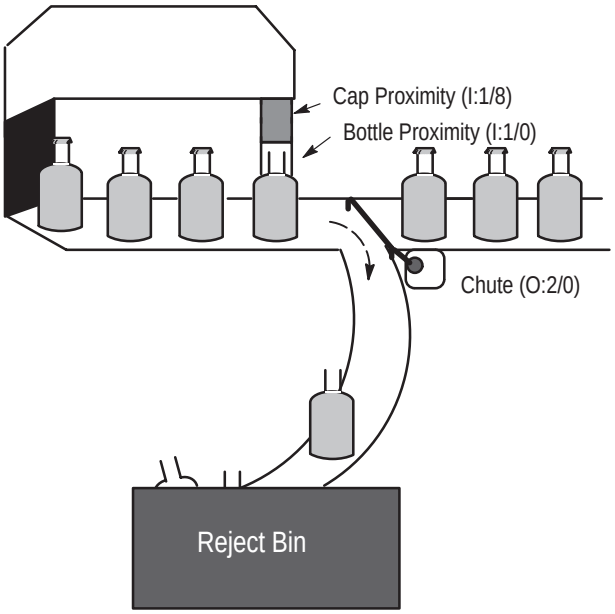
- **Return Mask (Word S:51)** – The Return Mask is updated immediately preceding entry into the DII subroutine. This value contains the bit map of the last bit transition that caused the interrupt. If more than one bit transitions in the same 100µs DII sample period, it is included in the return mask. This bit is cleared by the processor on exit from the DII subroutine. Use this value to validate the last interrupt transition that caused the input pattern to match the compare value. Or when dynamically reconfiguring (sequencing) the DII, use this value inside of your DII's subroutine to help determine/validate its position of the sequence.
- **Accumulator (Word S:52)** – The DII accumulator contains the number of counts that have occurred. When a count occurs and the accumulator is greater than or equal to the preset value, a DII interrupt is generated and the accumulator is cleared.

  For applications that measure the rate of incoming DII pulses while using a STI (Selectable Timed Interrupt), SLC 5/03 OS301 and above updates the DII accumulator prior to executing the first rung of the STI subroutine.

Discrete Input Interrupt Application Example

The following example shows how to use the Discrete Input Interrupt to control a high–speed application. In the example, the DII is used to ensure that all bottles exiting a filling and capping machine have their caps installed.

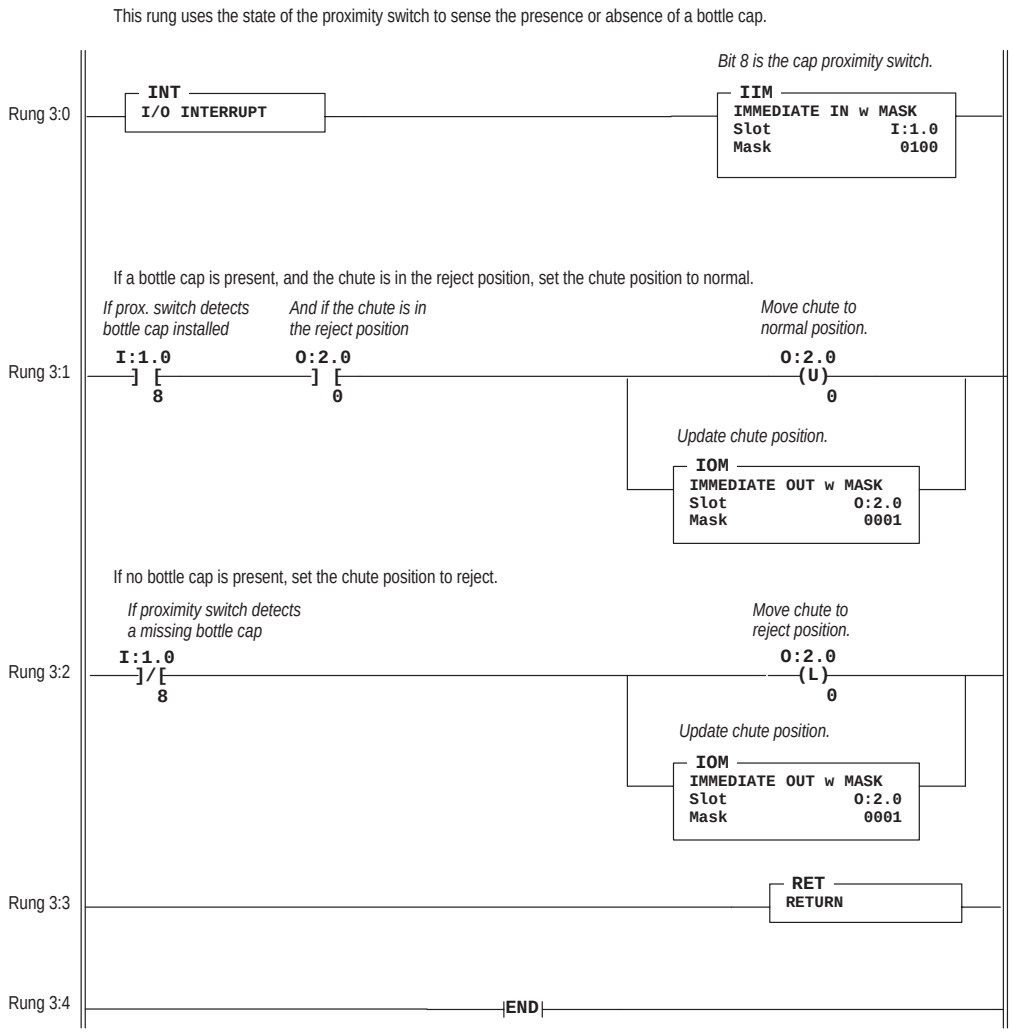
The bottle proximity switch is used as the DII input. When a bottle passes the proximity switch, the processor executes the DII subroutine. In the subroutine the processor reads the state of the cap proximity switch. If the cap is installed, the chute solenoid does not energize; allowing the bottle to continue down the line. If the cap is missing, the chute solenoid energizes, causing the defective bottle to divert down the chute and into the reject bin.



The following parameters are used to program the DII for the above application:

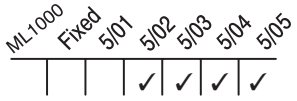
- S:33/8 Interrupt Latency Control Bit = 1
- S:46 File = 3
- S:47 Slot = 1
- S:48 Mask = 00000001
- S:49 Compare = 00000001
- S:50 Preset = 1

Ladder Diagram for the Bottling Application



Refer to appendix H for another application example using the DII to count pulses from an encoder.

I/O Interrupt Overview



This function allows a specialty I/O module to interrupt the normal processor operating cycle in order to scan a specified subroutine file. Interrupt operation for a specific module is described in the user's manual for the module.

Not all specialty I/O modules are capable of generating I/O interrupts. Refer to the user manual of the specific specialty I/O module to see if it supports this feature. For example, you cannot use a standard discrete I/O module to accomplish an I/O event–driven interrupt.

This section describes:

- I/O operation
- I/O interrupt parameters
- IID and IIE instructions
- RPI instruction
- INT instruction

Basic Programming Procedure for the I/O Interrupt Function

- When you are configuring the specialty I/O module slot with the programming device, make sure you program the "ISR" (interrupt subroutine) program file number (range 3 to 255) that you want the processor to execute when the module generates an interrupt. Specialty I/O modules that create interrupts should be configured in the lowest numbered I/O slots.
- Create the subroutine file that you have specified as the ISR number in the I/O module slot configuration.

Operation

When you restore your program and enter the REM Run mode, the I/O interrupt begins operation as follows:

1. The specialty I/O module determines that it needs servicing and generates an interrupt request to the SLC processor.

2. The processor is interrupted from what it is doing, and the specified interrupt subroutine file (ISR) is scanned.

3. When the ISR scan is completed, the specialty I/O module is notified. This informs the specialty I/O module that it is allowed to generate a new interrupt.

4. The processor resumes normal operation from where it left off.

Interrupt Subroutine (ISR) Content

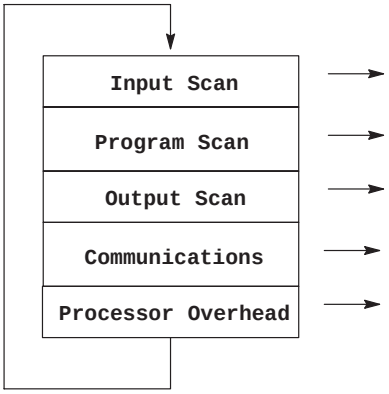
The Interrupt Subroutine (INT) instruction should be the first instruction in your ISR. This identifies the subroutine file as an I/O interrupt subroutine.

The ISR contains the rungs of your application logic. You can program any instruction inside an ISR except a TND, REF, or SVC instruction. IIM or IOM instructions are needed in an ISR if your application requires immediate update of input or output points. Terminate the ISR with an RET (return) instruction.

JSR stack depth is limited to 3. That is, you may call other subroutines to a level 3 deep from an ISR.

Interrupt Latency and Interrupt Occurrences

Interrupt latency is the interval between the I/O module's request for service and the start of the interrupt subroutine. I/O interrupts can occur at any point in your program, but not necessarily at the same point on successive interrupts. Interrupts can only occur between instructions in your program, inside the I/O scan (between slots), or between the servicing of communication packets. The following table shows the interaction between an interrupt and the processor operating cycle.



Events in the Processor Operating Cycle

SLC 5/02 I/O Interrupts	5/03 and Higher I/O Interrupts with Bit S:33/8 set	5/03 and Higher I/O Interrupts with Bit S:33/8 cleared
Between slot updates	Between word updates	Between slot updates
Between instruction updates	Between word updates	Between rung updates
Between slot updates	Between word updates	Between slot updates
Between communication packets	Between word packet updates	Between communication packets
At start and end	Between word updates	Between word upates

Note that ISR execution time adds directly to the overall scan time. During the latency period, the processor is performing operations that cannot be disturbed by the STI interrupt function. Latency periods are:

- SLC 5/02 interrupts are serviced within 2.4ms maximum.
- SLC 5/03 and higher processors – If an interrupt occurs while the processor is performing a multi-word slot update and your interrupt subroutine accesses that same slot, the multi-word transfer finishes to completion prior to performing the interrupt subroutine slot access. The Interrupt Latency Control bit (S:33/8) functions as follows:
  - When the bit is set (1) interrupts are serviced within the interrupt latency time. Refer to appendix D for more information on how to calculate the interrupt latency.
  - When S:33/8 is clear (0), user interrupts occur between rungs and I/O slot updates.

  The default state is cleared (0). To determine the interrupt latency with S:33/8 clear, you must calculate the execution time of each and every rung in your program.

Interrupt Priorities

Interrupt priorities are as follows:

SLC 5/02 Processor	SLC 5/03 and Higher Processors
1. Fault Routine	1. Fault routine
2. STI Subroutine	2. Discrete Input Interrupt (DII)
3. I/O Interrupt Subroutine (ISR)	3. STI Subroutine
	4. I/O Interrupt Subroutine (ISR)

An executing interrupt can only be interrupted by an interrupt having higher priority. The I/O interrupt cannot interrupt an executing fault routine, an executing DII subroutine, an executing STI subroutine, or another executing I/O interrupt subroutine. If an I/O interrupt occurs while the fault routine, DII, or STI subroutine is executing, the processor waits until the higher priority interrupts are scanned to completion. The I/O interrupt subroutine is then scanned.

Note *SLC 5/02 specific – It is important to understand that the I/O Pending bit associated with the interrupting slot remains* clear *during the time that the processor is waiting for the fault routine or STI subroutine to finish.*

Note *SLC 5/03 and higher processors – The I/O pending bit is always set when the interrupt occurs. You can examine the state of these bits within your higher priority interrupt routines.*

If a major fault occurs while executing the I/O interrupt subroutine, execution immediately switches to the fault routine. If the fault was recovered by the fault routine, execution resumes at the point that it left off in the I/O interrupt subroutine. Otherwise, the fault mode is entered.

If a DII interrupt occurs while executing the I/O interrupt subroutine, execution immediately switches to the DII subroutine. When the DII subroutine is scanned to completion, execution resumes at the point that it left off in the I/O interrupt subroutine.

If the STI timer expires while executing the I/O interrupt subroutine, execution immediately switches to the STI subroutine. When the STI subroutine is scanned to completion, execution resumes at the point that it left off in the I/O interrupt subroutine.

If two or more I/O interrupt requests are detected by the processor at the same instant, or while waiting for a higher or equal priority interrupt subroutine to finish, the interrupt subroutine associated with the specialty I/O module in the lowest slot number is scanned first. For example, if slot 2 (ISR 20) and slot 3 (ISR 11) request interrupt service at the same instant, the processor first scans ISR 20 to completion, then ISR 11 to completion.

Status File Data Saved

Data in the following words is saved on entry to the I/O interrupt subroutine and re-written upon exiting the I/O interrupt subroutine.

- S:0 Arithmetic flags
- S:13 and S:14 Math register
- S:24 Index register

I/O Interrupt Parameters

The I/O interrupt parameters below have status file addresses. They are described here and also in appendix B of this manual.

- **ISR Number** – Specifies the subroutine file number that will be executed when an I/O interrupt is generated by an I/O module. The ISR Numbers are not part of the status file, but they are part of the I/O configuration for each slot in the SLC system.
- **I/O Slot Enables (Words S:11 and S:12)** – These words are bit mapped to the 30 I/O slots. Bits S:11/1 through S:12/14 refer to slots 1 through 30. Bits S:11/0 and S:12/15 are reserved.

  The enable bit associated with an interrupting slot must be set when an interrupt occurs. Otherwise a major fault will occur. Changes made to these bits using the Data Monitor function take effect at the next end of scan.
- **I/O Interrupt Pending Bits (Words S:25 and S:26)** – These words are bit mapped to the 30 I/O slots. Bits S:25/1 through S:26/14 refer to slots 1 through 30. Bits S:25/0 and S:26/15 are reserved. The pending bit associated with an interrupting slot is set when the corresponding I/O slot interrupt enable bit is clear at the time of an interrupt request. It is cleared when the corresponding I/O event interrupt enable bit is set, or when an associated RPI instruction is executed. The pending bit for an executing I/O interrupt subroutine remains clear when the ISR is interrupted by a DII, STI, or fault routine.

  SLC 5/02 specific – Likewise, the pending bit remains clear if interrupt service is requested at the time that a higher or equal priority interrupt is executing (fault routine, STI, or other ISR).

  SLC 5/03 and higher processors – This bit is set if interrupt service is requested at the time a higher or equal priority interrupt is executing (fault routine, DII, STI, or other ISR).

- **I/O Interrupt Enables (Words S:27 and S:28)** – These words are bit mapped to the 30 I/O slots. Bits S:27/1 through S:28/14 refer to slots 1 through 30. Bits S:27/0 and S:28/15 are reserved. The enable bit associated with an interrupting slot must be set when the interrupt occurs to allow the corresponding ISR to execute. Otherwise the ISR will not execute and the associated I/O slot interrupt pending bit will be set.

  SLC 5/02 specific – Changes made to these bits using the data monitor function or ladder instruction take effect at the next end of scan.

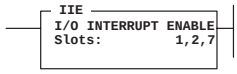
  SLC 5/03 and higher processors – Changes made to these bits using the data monitor function or ladder instruction take effect immediately.

- **I/O Interrupt Executing (Word S:32)** – This word contains the slot number of the specialty I/O module that generated the currently executing ISR. This value is cleared upon completion of the ISR, run mode entry, or upon power up. You can interrogate this word inside of your DII or STI subroutine or fault routine if you wish to know if these higher priority interrupts have interrupted an executing ISR. You may also use this value to discern interrupt slot identity when multiplexing two or more specialty I/O module interrupts to the same ISR.

I/O Interrupt Disable (IID) and I/O Interrupt Enable (IIE)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓	✓

These instructions are generally used in pairs to prevent I/O interrupts from occurring during time-critical or sequence-critical portions of your main program or subroutine. The I/O Event-Driven Interrupt function is used with specialty I/O modules capable of generating an interrupt.



Use these instructions together to create a zone in your main ladder program file or subroutine file in which I/O interrupts cannot occur. Both instructions take effect immediately upon execution. You must specify a subroutine to be executed upon receipt of such an interrupt.

SLC 5/02 specific – Setting/clearing the I/O interrupt enable bits (S:27 and S:28) with a programming device or standard instruction such as MVM takes effect at the END of the scan only.

SLC 5/03 and higher processors – Setting/clearing the I/O interrupt enable bits (S:27 and S:28) with a programming device or standard instruction such as MVM takes effect immediately.

IID Operation

When true, this instruction clears the I/O interrupt enable bits (S:27/1 through S:28/14) corresponding to the slots parameter of the instruction (slots 1, 2, 7 in the example above). Interrupt subroutines of the affected slots are not able to execute when an interrupt request is made. Instead, the corresponding I/O pending bits (S:25/1 through S:26/14) are set. The ISR is not executed until an IIE instruction with the same slot parameter is executed, or until the end of the scan during which you use a programming device to set the corresponding status file bit.

IIE Operation

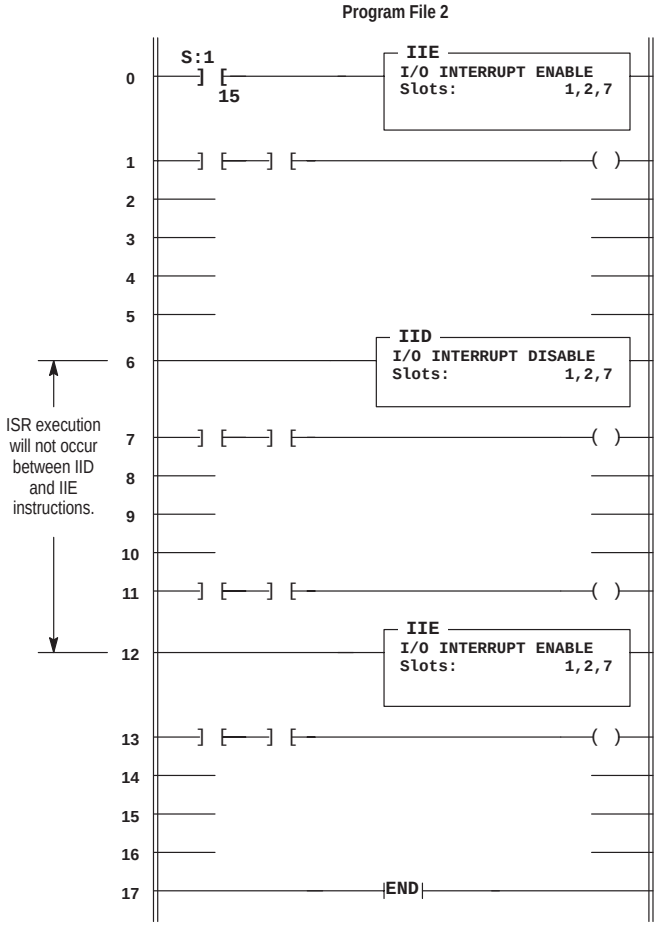
When true, this instruction sets the I/O interrupt enable bits (S:27/1 through S:28/14) corresponding to the slots parameter of the instruction (slots 1, 2, 7 in the example above). Interrupt subroutines of the affected slots regain the ability to execute when an interrupt request is made. If an interrupt was pending (S:25/1 through S:26/14) and the pending slot corresponds to the IIE slots parameter, the ISR associated with that slot executes immediately.

IID/IIE Zone Example

In the program below, slots 1, 2, and 7 are capable of generating I/O interrupts. The IID and IIE instructions in rungs 6 and 12 are included to avoid having I/O interrupt ISRs execute as a result of interrupt requests from slots 1, 2, or 7. This allows rungs 7 through 11 to execute without interruption.

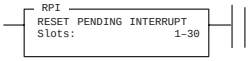
The first pass bit S:1/15 and the IIE instruction in rung 0 are included to insure that the I/O interrupt function is initialized following a power cycle. You should include a rung such as this any time your program contains an IID/IIE zone or an IID instruction.

The IID instruction in rung 6 clears the I/O interrupt enable bits associated with slots 1, 2, and 7 (S:27/1, S:27/2, and S:27/7). The IIE instruction in rung 12 sets these same bits. If an I/O interrupt is detected by the processor while the processor is executing rungs 7–11, the interrupt will be marked as pending. (S:25/1, S:25/2, and/or S:25/7 will be set.) All interrupts marked as pending are serviced upon execution of rung 12. The lowest numbered slot is serviced first when multiple pending bits are set.



Reset Pending Interrupt (RPI)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
			✓	✓	✓	✓



This instruction resets the pending status of the specified slots and informs the corresponding I/O modules that you have aborted their interrupt requests. This instruction is not required to configure a basic I/O interrupt application.

When true, this instruction clears the I/O pending bits (S:25/1 through S:26/14) corresponding to the slots parameter of the instruction. In addition, the processor notifies the specialty I/O modules in those slots that their interrupt request was aborted. Following this notice, the slot may once again request interrupt service. This instruction does *not* affect the I/O slot interrupt enable bits (S:27/1 through S:28/14).

Entering Parameters

Enter the I/O slot numbers (1 to 30) involved. Examples:

6	indicates slot 6
6,8	indicates slots 6 and 8
6–8	indicates slots 6, 7, and 8
1–30	indicates all slots

Interrupt Subroutine (INT)

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
✓			✓	✓	✓	✓

Use the INT instruction in I/O event-driven interrupt subroutines (ISRs) and STIs for identification purposes. Use of this instruction is optional.

This instruction has no control bits and is always evaluated as true. When used, the INT should be programmed as the first instruction of the first rung of the ISR.

13 SLC Communication Protocols

Use the information in this chapter to understand the differences in the communication protocols. The following protocols are supported:

- DH-485 Communication Protocol
- Data Highway Plus Communication Protocol
- DF1 Via RS-232 Communication Protocol
- ASCII Communication Protocol
- Ethernet Communication Protocol

In addition, operation of the Global Status Word (S:99) is provided.

Overview

DH-485 Protocol — The SLC 500 processors have a DH-485 channel that supports the DH-485 communication network. This network is a multi-master, token-passing network protocol capable of supporting up to 32 devices (nodes). This protocol allows:

- monitoring of data and processor status, along with program uploading and downloading of any device on the network from one location
- SLC processors to pass data to each other (peer-to-peer communication)
- operator interface devices on the network to access data from any SLC processor on the network

Data Highway Plus (DH+) Protocol — The Data Highway Plus protocol is used by the SLC 5/04 processor. This protocol is similar to DH-485, except that it can support up to 64 devices (nodes) and runs at faster communication (baud) rates.

DF1 Full-Duplex Protocol — DF1 Full-Duplex protocol (also referred to as DF1 point-to-point protocol) allows two devices to communicate with each other at the same time. This protocol allows:

- transmission of information across modems (dial-up, leased line, radio, or direct cable connections)
- communication to occur between Allen-Bradley products and third-party products

Note

You can connect to a DeviceNet network using the DeviceNet Interface (DNI), catalog number 1761-NET-DNI. The DNI provides a single DeviceNet connection point and a single RS-232 connection. The DNI is a DeviceNet to DF1 protocol conversion device that allows DF1 devices to communicate on DeviceNet.

For additional information on using the DNI, see the DeviceNet Interface (DNI) Overview, publication 1761-1.23, or the DeviceNet Interface (DNI) User Manual, publication 1761-6.5.

DF1 Half-Duplex Protocol (Master and Slave) — DF1 Half-Duplex protocol provides a multi-drop single master/multiple slave network capable of supporting up to 255 devices (nodes). This protocol also provides modem support and is ideal for SCADA (Supervisory Control and Data Acquisition) applications because of the network capability.

ASCII Protocol — The ASCII protocol provides connection to other ASCII devices, such as bar code readers, weigh scales, serial printers, and other intelligent devices.

Ethernet TCP/IP Protocol — The Ethernet protocol is used by the SLC 5/05 processor. Standard Ethernet, utilizing the TCP/IP protocol, is used as the backbone network in many office and industrial buildings. Ethernet is a local area network that provides communication between various devices at 10 Mbps. This network provides the same capabilities as DH+ or DH-485 networks, plus:

- SNMP support for Ethernet network management
- optional dynamic configuration of IP addresses using a BOOTP utility
- SLC 5/05 Ethernet data rate up to 40 times faster than SLC 5/04 DH+ messaging
- ability to message entire SLC 5/05 data files
- an unlimited number of nodes on a single network are possible compared to DH-485 (32) and DH+ (64)

The following table summarizes the communication options for the SLC 500 processor family.

Communications Protocol	Processor					
	Fixed	SLC 5/01	SLC 5/02	SLC 5/03	SLC 5/04	SLC 5/05
DH485 peer-to-peer	receive only	receive only	receive and initiate	receive and initiate	—	—
DH485 via RS232 port	—	—	—	receive and initiate①	receive and initiate①	receive and initiate①
DF1 via RS232 port (full-duplex or half-duplex master or slave)	receive only②	receive only②	receive only②	receive and initiate	receive and initiate	receive and initiate
ASCII via RS232 port	—	—	—	receive and initiate	receive and initiate	receive and initiate
Data Highway Plus (DH+)	receive only③	receive only③	receive only③	receive and initiate④	receive and initiate	receive and initiate④
Ethernet	—	—	—	—	—	receive and initiate

① If using 1747-AIC for isolation, connect to DH-485 network using 1747-PIC; if using 1761-NET-AIC for isolation, directly connect to DH-485 network with 1747-CP3 serial cable (or equivalent RS-232 null-modem cable).

② A 1747-KE or 1770-KF3 is required to bridge from DF1 (full-duplex or half-duplex slave only) to DH485.

③ A 1785-KA5 is required to bridge from DH+ to DH485.

④ Either a 1785-KA5 is required to bridge from DH+ to DH485 or the SLC-5/04's channel-to-channel passthru feature may be used to bridge between DH+ and DH485 or between DH+ and DF1 Full-Duplex (DH+ -to- DF1 Full-Duplex passthru available starting with OS401). Another option is to use the 1785-KE to bridge between DH+ and DF1 Full-Duplex or DH+ and a DF1 Half-Duplex Master/Slave network.

Note *The 1785-KA5 and 1785-KE modules require use of a 1771-series chassis and power supply.*

DH-485 Communication Protocol

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
	✓	✓	✓	✓	✓	✓

The DH-485 network offers:

- interconnection of 32 devices
- multi-master capability
- token passing access control
- the ability to add or remove nodes without disrupting the network
- maximum network length of 1219 m (4,000 feet)

DH-485 Network Protocol

The following section describes the protocol used to control message transfers on the DH-485 network. The protocol supports two classes of devices: initiators and responders. All initiators on the network get a chance to initiate message transfers. To determine which initiator has the right to transmit, a token passing algorithm is used.

DH-485 Token Rotation

A node holding the token can send valid packets onto the data link. The token hold parameter determines the number of transmissions (plus retries) each time the node receives the token.

After a node sends one message packet, it attempts to give the token to its successor by sending a “token pass” packet. If no network activity occurs, the initiator attempts to find a new successor.

The node address range for an initiator is 0-31. The node address range for all responders is 1-31. There must be at least one initiator on the network.

Note *The maximum address that the initiator searches for before wrapping to zero is the value in the configurable parameter “maximum node address.” The default value of this parameter is 31 for all initiators and responders.*

Note *The fixed, SLC 500 processors do not allow node address zero to be applied. If you attempt to apply a zero, node address one becomes the processor node address. Node address zero is reserved for a programming device, such as the Hand-Held Terminal (HHT) or personal computer running programming software.*

DH-485 Network Initialization

Network initialization begins when a period of inactivity exceeds the time of a “link dead timeout.” When the time for the “link dead timeout” is exceeded, usually the initiator with the lowest address claims the token.

Building a network begins when the initiator that claimed the token tries to pass the token to the successor node. If the attempt to pass the token fails, or if the initiator has no established successor (for example, when it powers up), it begins a linear search for a successor starting with the node above it. It will wrap to node 0 upon reaching its maximum node address value.

When the initiator finds another active initiator, it passes the token to that node, which repeats the process until the token is passed all the way around the network to the first node. At this point, the network is in a state of normal operation.

Software Considerations

Software considerations include the configuration of the network and the parameters that can be set to the specific requirements of the network. The following are major configuration factors that have a significant effect on network performance:

- number of nodes on the network
- addresses of those nodes
- baud rate
- maximum node address selection
- SLC 5/03 and higher – token hold factor
- maximum number of communicating devices

The following sections explain network considerations and describe ways to select parameters for optimum network performance (speed).

Number of Nodes

The number of nodes on the network directly affects the data transfer time between nodes. Unnecessary nodes (such as a second programming terminal that is not being used) slow the data transfer rate. The maximum number of nodes on the network is 32.

Setting Node Addresses

The best network performance occurs when node addresses start at 0 and are assigned in sequential order. SLC 500 processors default to node address 1. The node address is stored in the processor status file (S:15L). Processors cannot be node 0. Also, initiators such as personal computers should be assigned the lowest numbered addresses to minimize the time required to initialize the network.

If some nodes are connected on a temporary basis, do not assign addresses to them. Simply create nodes as needed and delete them when they are no longer required.

Setting Processor Baud Rate

The best network performance occurs at the highest baud rate. All devices must be at the same baud rate. The default DH-485 baud rate for SLC 500 devices is 19.2K baud. The baud rate is stored in the processor status file (S:15H).

Maximum Node Address Setting

The maximum node address parameter should be set as low as possible. This minimizes the amount of time used in soliciting successors when initializing the network. If all nodes are addressed in sequence from 0, and the maximum node address is equal to the address of the highest addressed node, the token rotation will improve by the amount of time required to transmit a solicit successor packet plus the slot timeout value.

Maximum Number of Communicating Devices

SLC 500 fixed and SLC 5/01 processors can be selected by two initiators, maximum at the same time. Using more than two initiators to select the same SLC 500 fixed and SLC 5/01 processors at the same time can cause communication timeouts.

DH-485 Configuration Parameters

When the system mode driver for channel 0 or channel 1 is DH-485 Master, the following parameters can be changed:

Parameter	Description
Diagnostic File	Reserved for future use.
Baud Rate	Toggles between the communication rate of 110, 300, 600, 1200, 2400, 4800, 9600, and 19200. The default is 19200.
Node Address	This is the node address of the processor on the DH-485 network. The valid range is 1–31. The default is 1.
Max Node Address	This is the maximum node address of an active processor. The valid range is 1–31. The default is 31.
Token Hold Factor	Determines the number of transactions allowed to make each DH-485 token rotation. Increasing this value allows your processor to increase its DH-485 throughput. This also decreases throughput to other processors on the DH-485 link. The valid range is 1–4. The default is 1. The SLC 5/01 and SLC 5/02 processors are factory set to 1.

The following devices use the DH-485 network:

Catalog Number	Description	Installation Requirement	Function	Publication
1746-BAS	BASIC Module	SLC Chassis	Provides an interface for SLC 500 devices to foreign devices. Program in BASIC to interface the 3 ports (2 RS-232 and 1 DH-485) to printers, modems, or the DH-485 network for data collection.	1746-6.1 1746-6.2 1746-6.3
1747-KE	DH-485/DF1 Interface Module	SLC Chassis	Provides a non-isolated DH-485 interface for SLC 500 to host computers over RS-232 using full- or half-duplex DF1 protocol. Enables remote programming using your programming software to an SLC 500 processor or the DH-485 network through modems. Ideal for low cost RTU/SCADA applications.	1747-6.12
1770-KF3	DH-485/DF1 Interface Module	Standalone “desktop”	Provides an isolated DH-485 interface for SLC 500 devices to host computers over RS-232 using full- or half-duplex DF1 protocol. Enables remote programming using your programming software to an SLC 500 processor or the DH-485 network through modems.	1770-6.5.18

Catalog Number	Description	Installation Requirement	Function	Publication
1784-KR	PC DH-485 Interface Module	IBM XT/AT Computer Bus	Provides an isolated DH-485 port on the back of the computer. When used with APS software, it improves communication speed and eliminates use of the Personal Interface Converter (1747-PIC). The Standard Driver allows you to write "C" programs for data acquisition applications.	1784-2.23 6001-6.5.5
1785-KA5	DH+/DH485 Gateway	(1771) PLC I/O Chassis	Provides communication between stations on the PLC-5 (DH+) and SLC 500 (DH-485) networks.	1785-6.5.5 1785-1.21
2760-RB	Flexible Interface Module	(1771) PLC Chassis	Provides an interface for SLC 500 (using protocol cartridge 2760-SFC3) to other A-B PLC processors and devices. Three configurable ports are available to interface with Bar Code, Vision, RF, Dataliners, and PLC systems.	2760-ND001
1747-DTAM, 2707-L8P1, -L8P2, -L40P1, -L40P2, -V40P1, -V40P2, -V40P2N, -M232P3, and -M485P3	DTAM, DTAM Plus, and DTAM Micro Operator Interfaces	Panel Mount	Provides electronic operator interface for SLC 500 processors.	1747-ND013 2707-800, 2707-803
2711-K5A2, -B5A2, -K5A5, -B5A5, -K5A1, -B5A1, -K9A2, -T9A2, -K9A5, -T9A5, -K9A1, and -T9A1	PanelView 550 and PanelView 900 Operator Terminals	Panel Mount	Provides electronic operator interface for SLC 500 processors.	2711-802, 2711-816

Data Highway Plus Communication Protocol

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
					✓	

Data Highway Plus implements peer-to-peer communication with a token-passing scheme to rotate link mastership among a maximum of 64 nodes. Since this method does not require polling, it helps provide time-efficient reliable data transport. The DH+ features:

- remote programming of PLC-2, PLC-3, PLC-5 and SLC 500 processors on your network
- direct connections to PLC-5 processors and industrial programming terminals
- easy re-configuration and expansion if you want to add more nodes later
- communication rates of 57.6K baud, 115.2K baud, or 230.4K baud

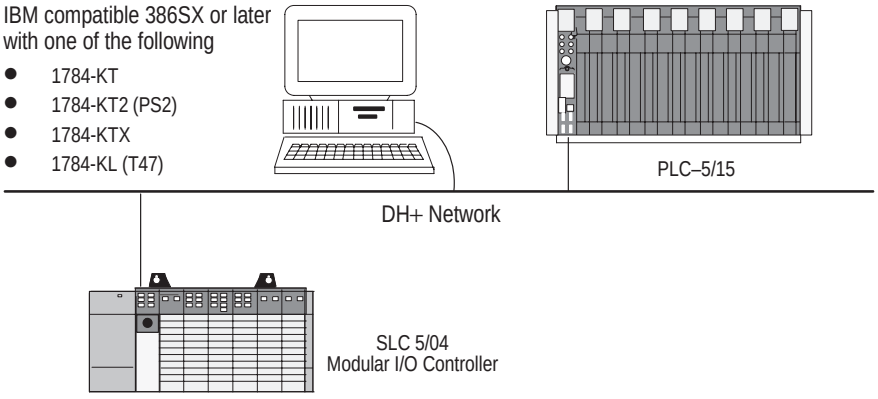
Important

A programming device, such as an IBM compatible PC, using a 1784-KT Communication Interface module does not operate faster than 57.6K baud. The 1784-KTXD can operate at all three communication rates.

The DH+ uses factory set timeouts to restart token-passing communication if the token is lost because of a defective node.

Example

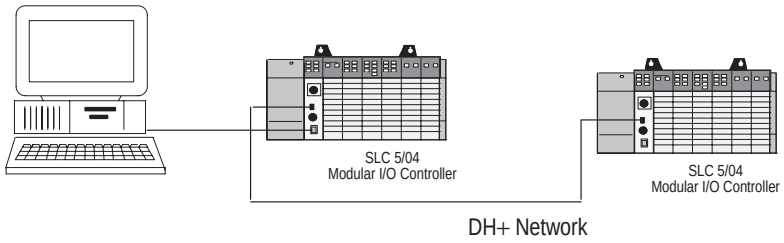
The example below shows the connectivity of an SLC 5/04 processor to a PLC-5 processor using the DH+ protocol. A communication rate of 57.6K baud is used.



Example

The example below shows a DH+ protocol using two SLC 5/04 controllers using the high baud rates of 115.2K baud or 230K baud.

Note *DH+ communication rates 115.2K baud and 230K baud are not available for the programming terminal unless a 1784-KTX card is used together with WINtelligent LINX or RSLinx. In the example below the programming terminal is connected to the serial port of the SLC 5/04 processor to enter the higher baud rate. This method uses the DF1 to DH+ passthru feature. For more information on passthru see chapter 8.*



DH+ Channel 1 Configuration Parameters (SLC 5/04 processors only)

When the system mode driver is DH+ for channel 1 the following parameters can be changed:

Parameter	Description
Baud Rate	Toggles between the communication rate of 57.6K, 115.2K, and 230.4K. The default is 57.6K. Make sure all devices on your DH+ link are configured for the same communication rate.
Node Address	Valid range is 0–77 octal. The default is 1.
Global Status Word Transmit Enable①	Toggles between 0 and 1. The default is 0.
Global Status Word Receive Enable①	Toggles between 0 and 1. The default is 0.

① This parameter is only available for SLC 5/04 (OS401) processors.

Global Status Word Overview

When a processor passes the DH+ token to the next node, it also sends a 16-bit word called the Global Status Word (GSW). Every node on the network sees the token pass message, but only the “next” node on the network accepts the token. However, all of the nodes on the network read the Global Status Word sent with each token pass and save it to memory. Each processor on the DH+ network has a table in memory to store Global Status Word(s) it receives from other nodes. In each SLC 5/04 processor’s status file, there is a designation for the:

- Global Transmit Word

  This word is located in memory at S:99. If, in your ladder program you move data to this memory location, it is transmitted every time the processor passes the DH+ token. Note that all other DH+ nodes see this data.

- Global Status File

  This file is located in memory at S:100 to S:163, representing one memory location for each of the 64 possible nodes on the DH+ network. As other nodes transmit Global Status information with their token passes, the SLC 5/04 processor collects this information and stores it in the Global Status File. Memory location S:100 corresponds to node #0 (octal), S:101 corresponds to node #2 (octal), and S:163 corresponds to node #77 (octal).

One word of every node’s Global Status File is updated each token pass. This can function as a high-speed broadcast message, useful for status passing and synchronization of processors.

If the Global Status Word Transmit Enable bit (S:34/3) and Global Status Word Receive bit (S:34/4) are never set, you can use the Global Status File (S:100 to S:163) for other storage uses. If these bits are used and then reset, the area in the System Status File is never altered by the SLC 5/04 processor, even after a power cycle to the processor.

Note *The System Status File must be at least 164 words in length for Global Status Word transmissions and receptions to take place. This means a user program for use with OS400 will not support the Global Status Word feature.*

S:34/3 Global Status Word Transmit Enable Bit (SLC 5/04 with OS401)

Transmission of the Global Status Word is enabled by setting bit S:34/3 in the status file. If this bit is set (1), the processor transmits the data in S:99 with every DH+ token pass. If this bit is not set (0), the processor passes the token and does not attach the Global Status Word. This bit is dynamically configurable and the default setting is zero. Keep the following guidelines in mind when using the Global Status Word Transmit Enable bit:

- If this bit is not set, the DH+ Token Pass transmitted out Channel 1 will contain no Global Status Word bytes
- If this bit is set, but the SLC 5/04 *is not* in RUN mode, REMote Run, or one of the three test modes, the DH+ Token Pass transmission will contain a 2-byte Global Status Word of 0x0000.
- If this bit is set and the SLC 5/04 *is in* RUN mode, REMote Run, or one of the three tests modes the DH+ Token Pass transmission will contain a 2-byte GSW equal to the value in S:99 (Global Status Word). The word is also placed in the 64-word Global Status File (S:100 to S:163) in the location corresponding to the DH+ node address associated with the SLC 5/04 processor.

  For example, if the SLC 5/04 processor is operating at octal address 22 (18 decimal), the transmitted GSW is written to word S:118.
- The word in the Global Status File corresponding to the SLC 5/04 processor's DH+ address will be set to 0x0000 if any thing is done to inhibit the transmission of the Global Status Word from S:99. This includes:
  - clearing S:34/3, Global Status Word Transmit Enable bit
  - placing the SLC 5/04 into a mode other than Run mode or Test mode
  - disabling Channel 1
  - an error occurring on the DH+ link to cause the Channel 1 LED to flash red or go solid red (This could be caused by a duplicate node address.)
  - not having an OS401 user program downloaded to the SLC 5/04 processor
- If S:34/3 is not set from the time the SLC 5/04 is powered up, the word corresponding to its DH+ address in the Global Status File will never be written to during the end-of-scan.

S:34/4 Global Status Word Receive Enable Bit (SLC 5/04 with OS401)

Receiving the Global Status Words of other processors on the network is enabled by setting bit S:34/4 in the status file. If this bit is set (1), the processor fills in the Global Status File with Global Status Words transmitted by other processors on the network. If this bit is not set (0), the processor ignores any Global Status Word activity on the network. This bit is dynamically configurable and the default setting is zero. Note that transmitting and receiving Global Status Words are independent of each other.

Keep the following guidelines in mind when using the Global Status Word Receive Enable bit:

- If this bit is not set, the Global Status File (S:100 to S:163) is not updated with Global Status Word information being passed on the link.
- An error occurring on the DH+ link to cause the Channel 1 LED to flash red or go solid red disables Global Status Word receptions. (This could be caused by a duplicate node address.)
- Global Status File (S:100-S:163) support is enabled when the following four conditions are met:
  - Channel 1 is configured for DH+ protocol communication
  - the System Status File is at least 164 words in length
  - the Global Status Word Receive Enable bit (S:34/3) is set
  - operation on the DH+ link is working (Channel 1 LED is green)
- The only processor mode that Global Status Word reception will not operate in is while downloading a program.

Note that all 164 words are updated during each end-of-scan. The following table describes possible states of the DH+ node address and the value written to the Global Status Word (S:99).

State of the DH+ Node Address	Value written into S:99 by the SLC 5/04 processor
Device is not active on the DH+ link	0x0000
Device is active on the DH+ link, but not sending GSW bytes in its Token Pass	0x0000
Device is active on the DH+ link and is sending 1 byte of GSW data in its Token Pass	High byte is set to 0x00; Low byte is set equal to 1 byte of GSW data
Device is active on the DH+ link and is sending 2 bytes of GSW data in its Token Pass	High byte is set equal to the second byte; Low byte is set equal to the first byte (or High and Low bytes are set equal to each other)
Device is active on the DH+ link and is sending 3 or 4 bytes of GSW data in its Token Pass	High byte is set equal to the second byte; Low byte is set equal to the first byte, and the third and fourth bytes are ignored

- If the Global Status File (S:100-S:163) is working and then Channel 1 is disabled, the entire Global Status File is zeroed out.
- If the Global Status File (S:100-S:163) is working and bit S:34/4 is reset, the entire Global Status File is zeroed out except for the one word corresponding to the Channel 1 DH+ node address.
- If the Global Status File (S:100-S:163) is working and then a DH+ link error occurs, the entire Global Status File is zeroed out. If the SLC 5/04 processor recovers from the error on its own, then the Global Status File updating resumes automatically.
- If the Global Status File (S:100-S:163) is working and then a user program with a System Status File of less than 164 words is downloaded, the SLC 5/04 processor detects this before any further updating of the Global Status File is attempted. In other words, no corruption of the user program results even if all other criteria are still met to support the GSW reception table feature.

Note

The SLC 5/04 processor maintains a working Global Status Word table regardless if Channel 1 DH+ Active Node Table operation is enabled, (by setting S:34/1). To view the Global Status Word table using your programming software, S:34/1 must be set in addition to meeting all of the above requirements.

PLC–5 to SLC 500 Communication Using PLC–2 Type MSG Commands

The SLC processors can send MSGs to a PLC-5 processor two ways:

If you are using this release:	Use this MSG instruction to communicate to a PLC-5 processor:
SLC 5/03 OS300	type 485CIF (PLC-2 emulation) For information, see this section.
SLC 5/03 OS301 and later SLC 5/04 SLC 5/05	• type PLC-5 (the preferred method) • type 485CIF (PLC-2 emulation) For information, see page 13–18

Program a PLC-5 message instruction as type PLC-2 when accessing an SLC 500 processor. Newer enhanced PLC-5 processors also support SLC typed read and write messages in their message instruction.

PLC-2 Unprotected Reads and Writes are not really implemented as “unprotected” in the SLC processor. They are subject to the SLC’s file protection schemes. For instance, they are rejected if a download is in process or the Common Interface File, (CIF) is already open by another device. These types of read and write commands are somewhat “universal” in that they are implemented in many other Allen-Bradley programmable controllers.

The CIF is actually like any of the other SLC data files except that it is designated as the target file for all PLC-2 Unprotected Read and Unprotected Write commands that are received by the SLC. It is always File #9. The CIF can be defined as Bit, Integer, Timer, Counter, or Control Data types. However, only Bit or Integer files should be used to make addressing easier.

You cannot use the SLC 5/02 message instruction to send a message through a 1785-KA5 module. However, you can use the SLC 5/03 message instruction to send a message to the 1785-KA5 module. The SLC 5/03 processor has the ability to respond to data read/write requests when the 1785-KA5 is in the “router mode.” The fixed SLC 500, SLC 5/01, and SLC 5/02 processors cannot respond to data read/write requests. When the 1785-KA5 is in the gateway mode, all SLC 500 processors can respond to Data Highway Plus data read/write requests.

Note

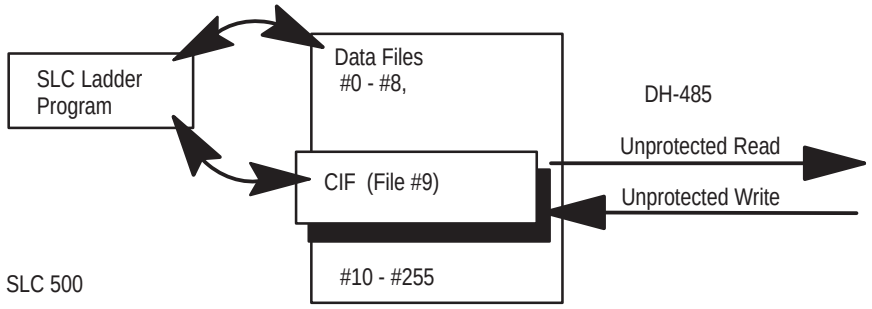
File #9 must be created and defined at the time that the SLC is programmed. File #9 must also be made large enough to include the Unprotected Read and Write addressing space. Otherwise, all Unprotected Reads and Writes will be rejected by the SLC.

How the PLC-5 Processors Address Data

When programming a PLC-2 type of MSG instruction on the PLC-5, the "Destination Address" is entered in octal. The PLC-5 processor automatically translates the octal address to a *byte* address by doubling the decimal equivalent. Therefore, 010_8 becomes 16 and 177_8 becomes 254. You cannot enter an octal address less than 010_8 in a PLC-2 type PLC-5 Message instruction.

Using the SLC 500 CIF File (PLC-2 Emulation)

The CIF can be thought of as a data buffer between all the other SLC data files and the DH-485 channel. The SLC must be programmed, using ladder logic, to transfer data between the CIF and the other data files as shown here.



The CIF can be managed by designating areas to be written to and areas to be read from. If it is desired to know when data has changed in the CIF, use ladder logic to program handshaking bits in your CIF data.

Note

Although the format of the Unprotected Reads and Writes is the same as used in other PLC processors, the implementation of the address parameter is different. In Allen-Bradley's PLC products, the address is interpreted as a byte address. In some SLC 500 products, the address is interpreted as a word address.

- *The SLC 500 and SLC 5/01 processors use* word *addressing exclusively.*
- *The SLC 5/02, prior to Series C FRN 3 processors, also use* word *addressing exclusively.*
- *The SLC 5/02, SLC 5/03, SLC 5/04, and SLC 5/05 processors have a selection bit, S:2/8, which allows selection of either* word *or* byte *addressing.*
- *The DTAM for the SLC uses* word *addressing exclusively.*

Programming to Handle the Word/Byte Addressing Differences

SLC 500 processors use **word** addressing while PLC-5 processors use **byte** addressing. One byte in the PLC-5 processors equals two words in the SLC 500 processor.

The following section describes the differences between word and byte addressing when sending messages to/from a PLC-5 processor via PLC-2 commands.

Sending a PLC-2-Type Message to a PLC-5 Processor by Using SLC "Word" Addressing (S:2/8 = 0)

The PLC-5 Message instruction's PLC-2 type octal "Destination Address" must be between 010_8 and 177_8. This range corresponds to *word* 16 through *word* 254 (even words only) when S:2/8 equals zero.

Sending a PLC-2 Type Message to a PLC-5 Processor Using SLC "Byte" Addressing (S:2/8 = 1)

Note *The byte addressing mode is selected in the SLC by setting bit S:2/8 to 1. The default is S:2/8 = 0 for* word *addressing. This selection bit is* not *available in the SLC fixed or SLC 5/01 processors. This setting applies to the offset byte/word.*

The PLC-5 Message instruction's octal "Destination Address" must be between 010_8 and 377_8. This range corresponds to *word* 8 through *word* 254 when S:2/8 equals 1.

PLC-5 MSG Destination Address (Octal)	SLC Address	
	Word Mode (S:2/8=0)	Byte Mode (S:2/8=1)
010	N9:16	N9:8
011	N9:18	N9:9
...	...	...
177	N9:254	N9:127
200		N9:128
...		...
377		N9:255

The maximum value for the PLC-5 processor PLC-2-type instruction "Size in Elements" parameter is 41 for a SLC 5/02 processor and 110 for a SLC 5/03 processor (assuming 1 word elements).

Example - Sending a PLC-2 Type Message to a PLC-5 Processors Using “word” addressed SLC processors (S:2/8 = 0)

As an example, write 10 words from N7 in a PLC-5 to an SLC 5/02:

1. Set up the source address in the message instruction as N7:0.
2. Set the “Size in elements” to 10.
3. Set up the “Command Type” as “PLC-2 Unprotected Write.”
4. Set up the “Destination Address” as 010_8. This corresponds to the SLC address, N9:16.

Since 10 words are written, make sure that the N9 file in the SLC is created to at least N9:25.

It is assumed that the MSG instruction will be set up for a remote destination, since there must be a bridge between the PLC-5 and the SLC 5/02, such as a 1784-KA5 (in gateway mode) linking a DH+ and a DH-485 network.

Example - Sending a PLC-2 Type Message to a PLC-5 Processor using “byte” addressed SLC processors (S:2/8 = 1)

As an example, write 10 words from N7 in a PLC-5 to an SLC 5/02:.

1. Set up the source address in the message instruction as N7:0.
2. Set the “Size in elements” to 10.
3. Set up the “Command Type” as “PLC-2 Unprotected Write.”
4. Set up the “Destination Address” as 010_8. This corresponds to the SLC address, N9:7.

Since 10 words will be written, make sure that the N9 file in the SLC is created to at least N9:17.

SLC 5/03, SLC 5/04, and SLC 5/05 Processors to PLC-5 Communication Using SLC 500 or PLC-5 MSG Commands

The SLC 5/03 (OS301), SLC 5/04, and SLC 5/05 processors support PLC-5 type MSG commands. This eliminates having to program PLC-2 type MSGs.

When you want to access:	Program the MSG instruction as:
PLC-5 processors 1785-KA5 communication interface module	type PLC-5 MSG
SLC 5/03 SLC 5/04 SLC 5/05	type SLC 500

The SLC 5/03 (OS301), SLC 5/04, and SLC 5/05 processors accept PLC-5 type MSG commands to read and write the status, bit, timer, counter, control, integer, floating point, string, and ASCII data files. However, the SLC 5/03 (OS301), SLC 5/04, and SLC 5/05 processors do not accept PLC-5 type MSG commands to read or write to/from input and output files due to the difference between the PLC-5 processor's chassis/group addressing structure and the SLC 500's slot/word addressing structure. Additionally, the PLC-5 processor currently does not accept any SLC 500 MSG commands.

When programming a PLC-5 type MSG instruction, the source and destination data types should match. For consistency in the data transfer, we recommend that the destination and source data types match when you are transferring data between the PLC-5 processors and SLC 5/03 (OS301), SLC 5/04, and SLC 5/05 processors.

When programming an SLC MSG instruction, the source and destination data types do not have to match.

The destination data type determines the number of words per element to transfer. For example, T4:0 destination and a N7:0 source with a length of 3 results in a transfer of 9 integer words due to a Timer element size of 3 words per element.

DF1 Via RS-232 Communication Protocol

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

The SLC 5/03, SLC 5/04, and SLC 5/05 processors support DF1 Full-Duplex protocol and DF1 Half-Duplex master/slave protocol via the RS-232 connection to a host computer (using DF1 channel). Refer to *DF1 Protocol and Command Set Reference Manual*, publication 1770-6.5.16, for more information on these communication protocols.

For more information about using the SLC 500 processors in SCADA applications, see the:

- *SCADA System Selection Guide*, publication AG-2.1
- *SCADA System Application Guide*, publication AG-6.5.8

DF1 Full-Duplex Protocol

DF1 Full-Duplex protocol (also referred to as DF1 point-to-point protocol) is provided for applications where RS-232 point-to-point communication is required. This type of protocol supports simultaneous transmissions between two devices in both directions. You can use channel 0 as a programming port, or as a peer-to-peer port using the MSG instruction.

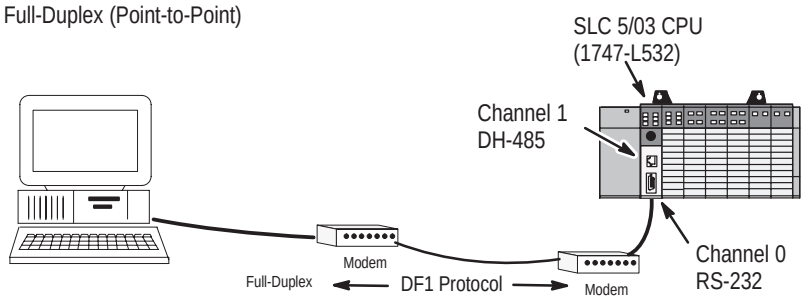
In full-duplex mode, the SLC 5/03 (or higher) processor can send and receive messages. When the processor receives messages, it acts as an end device — a device that stops the transmission of data packets. The processor ignores the destination and source addresses received in the data packets. However, the processor exchanges these addresses in the reply that it transmits in response to any command data packet that it has received.

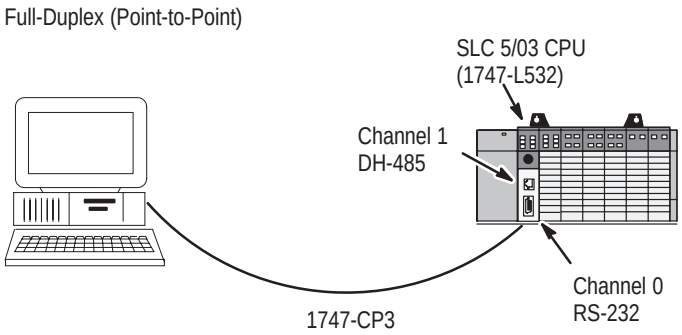
If you use a modem with DF1 channel 0 in the full-duplex mode, it must be capable of operating in full-duplex mode. Typically, a dial-up modem is used for communication over telephone lines.

DF1 Full-Duplex Channel 0 Configuration Parameters

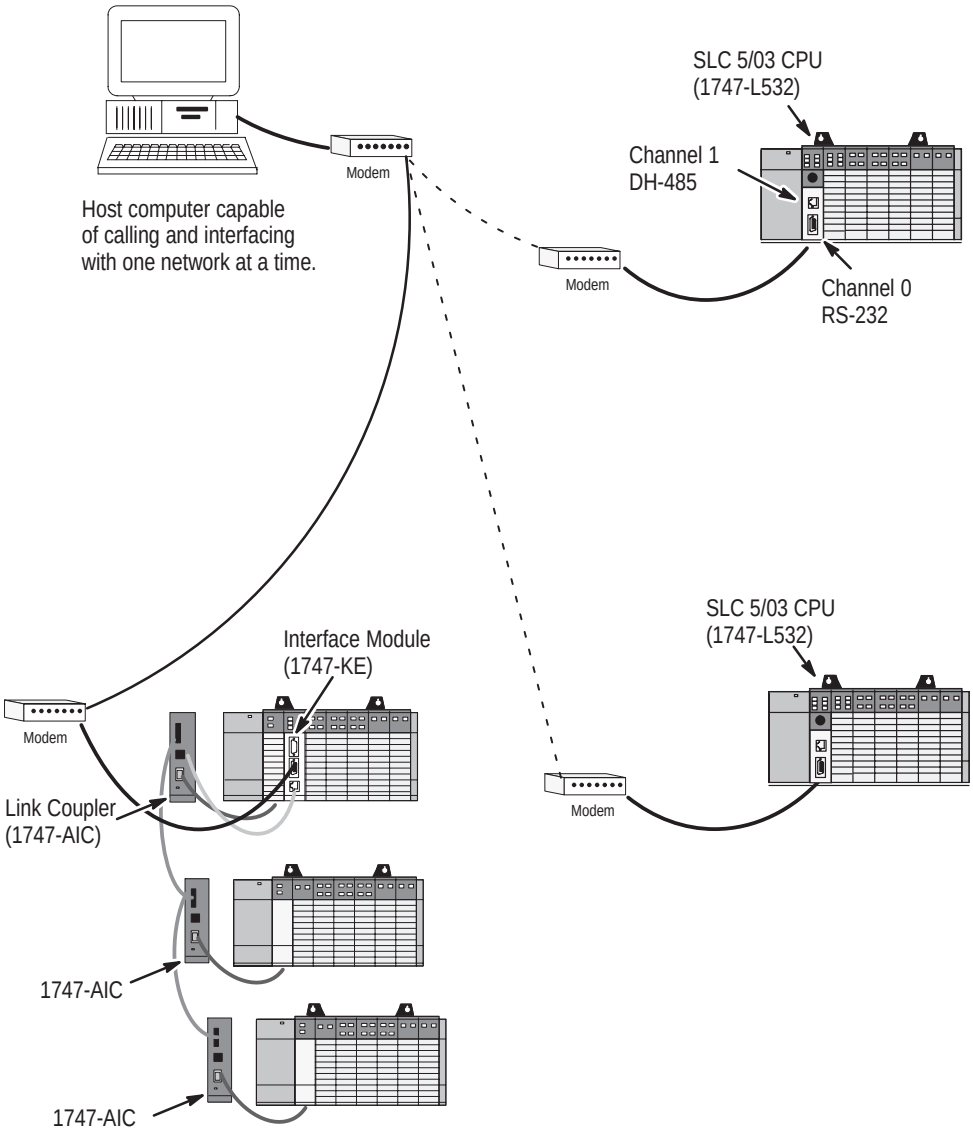
When the system mode driver is a DF1 Full-Duplex for channel 0 the following parameters can be changed:

Parameter	Description
Diagnostic File	Reserved for future use.
Baud Rate	Toggles between the communication rate of 110, 300, 600, 1.2K, 2.4K, 4.8K, 9.6K, and 19.2K (additional rate of 38.4K for SLC 5/05 only). The default is 1.2K for SLC 5/03 and SLC 5/04, and 19.2K for the SLC 5/05.
Parity	Toggle between None and Even. The default is None.
Stop Bits	Toggles between 1, 1.5, and 2. The default is 1.
Duplicate Packet Detection	Toggles between Disabled and Enabled. The default is Enabled.
Error Detection	Toggles between CRC and BCC. The default is CRC.
ACK Timeout	Valid range is 2–65535 (in 20 ms increment) The default is 50.
NAK Retries	Valid range is 0–255. The default is 3.
ENQ Retries	Valid range is 0–255. The default is 3.
Control Line	Toggles between No Handshaking and Full-Duplex modem. The default is No Handshaking.
Embedded Responses	Toggles between Enabled and Auto-Detect. The default is Enabled.
Source ID	Specify the address of the sender in this field. Valid range is 0–254. The default is 9.





Full-Duplex (Network)



This configuration allows the host to call more than one remote network. Each remote network can consist of up to 31 SLC nodes.

DF1 Half-Duplex Master/Slave Protocol

DF1 Half-Duplex Master/Slave protocol provides a multi-drop single master/multiple slave network. In contrast to DF1 full-duplex, communication takes place in one direction at a time. You can use channel 0 as a programming port, or as a peer-to-peer port using the MSG instruction.

The master device initiates all communication by “polling” each slave device. The slave device may only transmit data packets when it is polled by the master. It is the master’s responsibility to poll each slave on a regular and sequential basis to collect data. During a polling sequence, the master polls a slave repeatedly until the slave indicates that it has no more data packets to transmit. The master then transmits the data packets for that slave.

Several Allen-Bradley products support half-duplex master protocol. They include the enhanced PLC-5 processors, SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors. WINtelligent Linx and RSLinx (V2.0 and higher) software also support half-duplex master protocol.

Typically, the master keeps two separate tables -– one for online slaves and one for offline slaves. The online slaves are polled on a regular basis. The offline slaves are polled occasionally to see if they have come back online.

A master device supports routing of data packets from one slave to another.

DF1 half-duplex supports up to 255 slave devices (address 0 to 254) with address 255 reserved for master broadcasts. Either half-duplex or full-duplex modem types can be used for DF1 half-duplex network. The SLC 5/03, SLC 5/04, and SLC 5/05 support broadcast reception, but cannot initiate a broadcast command.

DF1 Half-Duplex Slave Channel 0 Configuration Parameters

When the system mode driver is DF1 Half-Duplex Slave for channel 0 the following parameters can be changed:

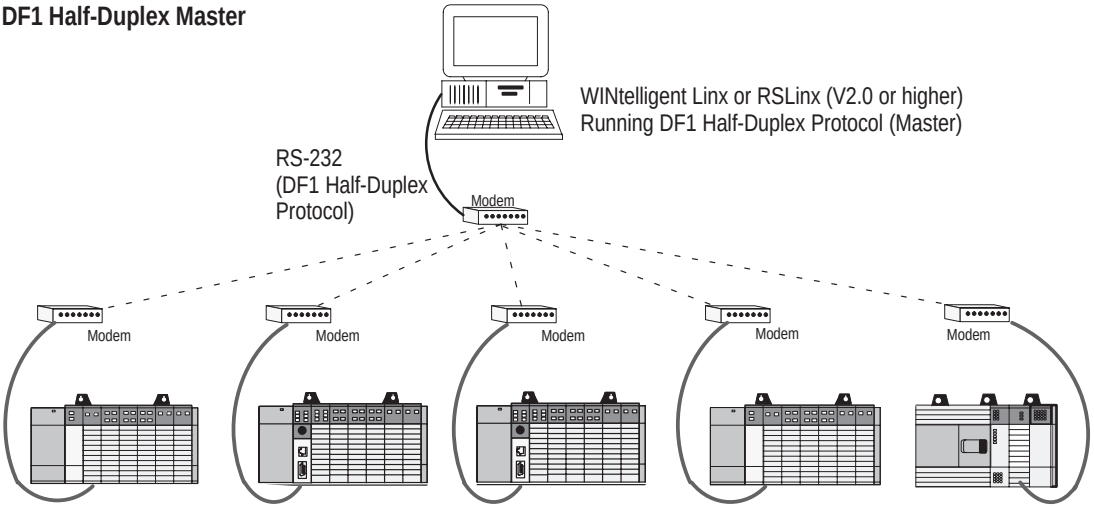
Parameter	Description
Diagnostic File	Reserved for future use.
Baud Rate	Toggles between the communication rate of 110, 300, 600, 1.2K, 2.4K, 4.8K, 9.6K, and 19.2K (additional rate of 38.4K for SLC 5/05 only). The default is 1.2K for SLC 5/03 and SLC 5/04, and 19.2K for the SLC 5/05.
Parity	Toggles between None and Even. The default is None.
Stop Bits	Toggles between 1, 1.5, and 2. The default is 1.
Station Address	The valid range is 0–254 decimal. The default is 1.
Duplicate Packet Detection	Toggles between Enabled and Disabled. The default is Enabled.
Error Detection	Toggles between CRC and BCC. The default is CRC.
RTS Off Delay	Allows you to select the RTS off delay value in increments of 20 ms. The valid range is 0–65535. The default is 0.
RTS Send Delay	Allows you to select the RTS send delay value in increments of 20 ms. The valid range is 0–65535. The default is 0.
Poll Timeout	Allows you to select the master poll timeout value in increments of 20 ms. The default is 50. The valid range is 0–65535.
Pre-send Time Delay	Allows you to select the RTS pre-transmit time delay in increments of 20 ms. The valid range is 0–65535. The default is 0.
Message Retries	Allows you to select the message retries value. The valid range is 0–255. The default is 3.
Control Line	Toggles between No Handshaking, Half-Duplex with continuous carrier, and Half-Duplex without continuous carrier. The default is No Handshaking.
EOT Suppression	Toggles between Yes and No. The default is No.

DF1 Half-Duplex Master Channel 0 Configuration Parameters

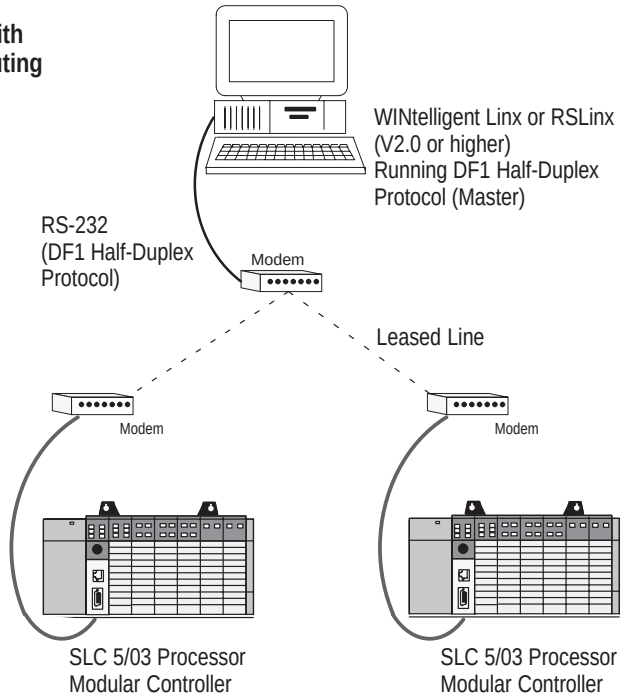
When the system mode driver is DF1 Half-Duplex Master for channel 0, the following parameters can be changed:

Parameter	Description
Diagnostic File	Reserved for future use.
Baud Rate	Toggles between the communication rate of 110, 300, 600, 1.2K, 2.4K, 4.8K, 9.6K, and 19.2K (additional rate of 38.4K for SLC 5/05 only). The default is 1.2K for SLC 5/03 and SLC 5/04, and 19.2K for the SLC 5/05.
Parity	Toggles between None and Even. The default is None.
Stop Bits	Toggles between 1, 1.5, and 2. The default is 1.
Station Address	The valid range is 0–254 decimal. The default is 1.
Duplicate Packet Detection	Toggles between Enabled and Disabled. The default is Enabled.
Error Detection	Toggles between CRC and BCC. The default is CRC.
ACK Timeout	Allows you to select the ACK timeout value in increments of 20 ms. The valid range is 0–65535. The default is 50.
RTS Off Delay	Allows you to select the RTS off delay value in increments of 20 ms. The valid range is 0–65535. The default is 0.
Message Retries	Allows you to select the message retries value. The valid range is 0–255. The default is 3.
RTS Send Delay	Allows you to select the RTS send delay value in increments of 20 ms. The valid range is 0–65535. The default is 0.
Pre-send Time Delay	Allows you to select the RTS pre-transmit time delay in increments of 20 ms. The valid range is 0–65535. The default is 0.
Control Line	Toggles between No Handshaking, Full-Duplex Modem, and Half-Duplex Without Continuous Carrier.
Polling Mode	Toggles between Message Based (Do Not Allow Slave to Initiate Messages), Message Based (Allow Slave to Initiate Messages), Standard (Single Message Transfer Per Node Scan), and Standard (Multiple Message Transfer Per Node Scan).
Priority Polling – Low	Allows you to select the Priority Polling Range Low Address. The valid range is 0–255.
Normal Polling – Low	Allows you to select the Normal Polling Range Low Address. The valid range is 0–255.
Priority Polling – High	Allows you to select the Priority Polling Range High Address. The valid range is 0–254.
Normal Polling – High	Allows you to select the Normal Polling Range High Address. The valid range is 0–254.
Reply Message Wait Time	Allows you to select the Reply Message Wait Time setting in increments of 20 ms. The valid range is 0–65535.
Normal Poll Group Size	Allows you to select the Normal Poll Group Size. The valid range is 0–255.

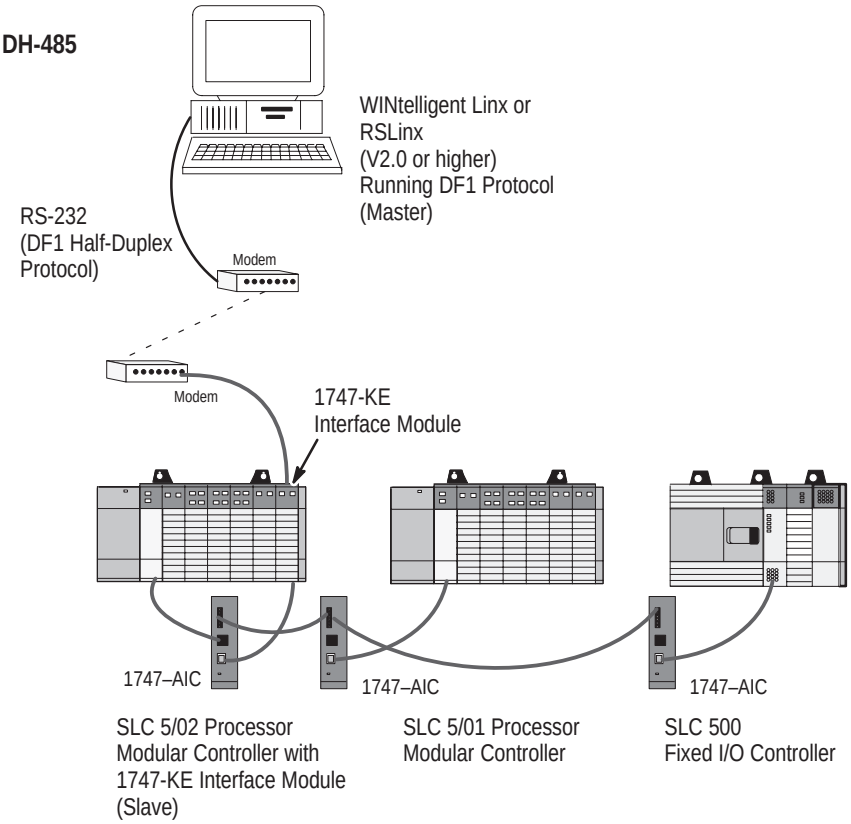
DF1 Half-Duplex Master



DF1 Half-Duplex with Slave-to-Slave Routing



DF1 Half-Duplex to DH-485 Multi-Drop Link



Considerations When Communicating as a DF1 Slave on a Multi–drop Link

When communication is between either your programming software and an SLC processor or between two SLC processors via a slave-to-slave connection on a larger multi-drop link, the devices depend on a DF1 Master to give each of them polling permission to transmit in a timely manner. As the number of slaves increase on the link (up to 254), the time between when your programming software or the SLC processor is polled also increases. This increase in time may become larger if you are using low baud rates.

As these time periods grow, the following values may need to be changed to avoid loss of communication:

- programming software - poll timeout and reply timeout values
- SLC processor - poll timeout and edit resource/file owner timeout values. If you are using MSG instructions between SLC processors, the MSG timeout value shown in the control block may also need to be changed for reliable slave-to-slave communication on the multi-drop network.

Using Modems that Support DF1 Communication Protocols

The types of modems that you can use with SLC processors include dial-up phone modems, leased-line modems, radio modems and line drivers. For point-to-point full-duplex modem connections, use DF1 full-duplex protocol. For point-to-multipoint modem connections, use DF1 half-duplex master and slave protocols. In this case, one (and only one) of the other devices must be configured for DF1 half-duplex master protocol. *Do not attempt to use DH-485 protocol through modems under any circumstance.*

Dial-Up Phone Modems

Dial-up phone line modems support point-to-point full-duplex communications. Normally an SLC processor, on the initiating or receiving end of the dial-up connection, will be configured for DF1 full-duplex protocol with the control line parameter set for “Full-Duplex Modem”. See page 13–31 for details on the operation of the RS-232 modem control signals when “Full-Duplex Modem” is selected.

When an SLC processor is the initiator of the dial-up connection, use one of the ASCII write instructions to send out the “AT” dial-up string (for example: ATDT 555-1212). The status file modem lost bit (S:5/14) provides the feedback that the connnection has been successfully made. To hang up the connection, use the ASCII AHL instruction to temporarily lower the DTR signal.

Leased-Line Modems

Leased-line modems are used with dedicated phone lines that are typically leased from the local phone company. The dedicated lines may be in a point-to-point topology supporting full-duplex communications between two modems or in a point-to-multipoint topology supporting half-duplex communications between three or more modems. In the point-to-point topology, configure the SLC processor for DF1 full-duplex protocol with the control line parameter set to “Full-Duplex Modem”. In the point-to-multipoint topology, configure the SLC processors for DF1 half-duplex master or slave protocol with the control line parameter set to “Half-Duplex Modem without Continuous Carrier”. See page 13–31 for details on the operation of the RS-232 modem control signals when “Half-Duplex Modem without Continuous Carrier” is selected.

Radio Modems

Radio modems may be implemented in a point-to-point topology supporting either half-duplex or full-duplex communications, or in a point-to-multipoint topology supporting half-duplex communications between three or more modems. In the point-to-point topology using full-duplex radio modems, configure the SLC processors for DF1 full-duplex protocol. In the point-to-point topology using half-duplex radio modems, or point-to-multipoint topology using half-duplex radio modems, configure the SLC processors for DF1 half-duplex master or slave protocol. If these radio modems require RTS/CTS handshaking, configure the control line parameter to “Half-Duplex Modem without Continuous Carrier”.

Line Drivers

Line drivers, also called short-haul “modems”, do not actually modulate the serial data, but rather condition the electrical signals to operate reliably over long transmission distances (up to several miles). Allen-Bradley’s AIC+ Advanced Interface Converter is a line driver that converts an RS-232 electrical signal into an RS-485 electrical signal, increasing the signal trasmission distance from 50 to 4000 feet. In a point-to-point line driver topology, configure the SLC processor for DF1 full-duplex protocol. In a point-to-multipoint line driver topology, configure the SLC processors for DF1 half-duplex slave protocol. If these line drivers require RTS/CTS handshaking, configure the control line parameter to “Half-Duplex Modem without Continuous Carrier”.

Modem Control Line Operation in SLC 5/03, SLC 5/04 and SLC 5/05 Processors

The following explains the operation of the SLC 5/03, SLC 5/04, and SLC 5/05 processors when you configure the RS232 channel for the following applications.

DF1 Full–Duplex

When you configure the SLC 5/03, SLC 5/04, and SLC 5/05 processors for full-duplex DF1, the following control line operation takes effect:

No Handshaking Selected — DTR is always active and RTS is always inactive. Receptions and transmissions take place regardless of the states of DSR, CTS, or DCD inputs. This selection should only be made when the SLC 5/03, SLC 5/04 and SLC 5/05 processors are directly connected to another DTE device.

Full-Duplex Modem Selected — DTR and RTS are always active except at the following times. If DSR goes inactive, both DTR and RTS are dropped for 1 to 2 seconds then reactivated. The modem lost bit (S:5/14) is turned on immediately. While DSR is inactive, the state of DCD is ignored. Neither receptions nor transmissions are performed.

If DCD goes inactive while DSR is active, then receptions are not allowed. If DCD remains inactive for 9 to 10 seconds, then DTR is set inactive until DSR goes inactive. At this point, the modem lost bit is also set. If DSR does not go inactive, then DTR is raised again in 5 to 6 seconds.

Transmission requires all three inputs (CTS, DCD, and DSR) to be active. Whenever DSR and DCD are both active, the modem lost bit is reset.

DF1 Half-Duplex Slave

When you configure the SLC 5/03, SLC 5/04, and SLC 5/05 processors for DF1 half-duplex slave, the following control line operation takes effect:

No Handshaking Selected — DTR is always active and RTS is always inactive. Receptions and transmissions take place regardless of the states of DSR, CTS, or DCD inputs. This selection should only be made when the processor is directly connected to another DTE device.

Half-Duplex Modem with Continuous Carrier Selected — DTR is always active and RTS is only activated during transmissions (and any programmed delays before or after transmissions). The handling of DCD and DSR are exactly the same as with Full-Duplex Modem. Transmissions require CTS and DSR to be active.

Half-Duplex Modem without Continuous Carrier Selected — This is exactly the same as Half-Duplex Modem with Continuous Carrier except monitoring of DCD is not performed. DCD is still required for receptions but is not required for transmissions. Transmissions still require CTS and DSR. The modem lost bit will only be set when DSR is inactive.

DF1 Half-Duplex Master

When you configure the SLC 5/03, SLC 5/04, and SLC 5/05 processors for DR1 half-duplex master, the following control line operation takes effect:

No Handshaking Selected — DTR is always active and RTS is always inactive. Receptions and transmissions take place regardless of the states of DSR, CTS, or DCD inputs. This selection should only be made when the processor is directly connected to another DTE device.

Full-Duplex Modem Selected — DTR and RTS are always active except at the following times. If DSR goes inactive, both DTR and RTS are dropped for 1 to 2 seconds then reactivated. The modem lost bit (S:5/14) is turned on immediately. While DSR is inactive, the state of DCD is ignored. Neither receptions nor transmissions are performed.

If DCD goes inactive while DSR is active, then receptions are not allowed. If DCD remains inactive for 9 to 10 seconds, then DTR is set inactive until DSR goes inactive. At this point, the modem lost bit is also set. If DSR does not go inactive, then DTR is raised again in 5 to 6 seconds.

Transmission requires all three inputs (CTS, DCD, and DSR) to be active. Whenever DSR and DCD are both active, the modem lost bit is reset.

Half-Duplex Modem without Continuous Carrier Selected — DTR is always active and RTS is only active during transmissions (and any programmed delays before and after transmissions). The processor does not monitor DCD.

If DSR goes inactive, RTS is dropped. The modem lost bit (S:5/14) is turned on immediately. While DSR is inactive, neither receptions nor transmissions are performed.

Transmission requires two inputs, CTS and DSR, to be active. Whenever DSR is active, the modem lost bit is resest.

RTS Send Delay and RTS Off Delay Parameters

Through your programming software, the parameters RTS Send Delay and RTS Off Delay give you the flexibility of selecting modem control during transmissions. These parameters only apply when you select half-duplex modem with or without continuous carrier.

For use with half-duplex modems that require extra time to “key up” their transmitter even after they have activated CTS, the RTS Send Delay specifies in 20 millisecond increments the amount of delay time after activating RTS to wait before checking to see if CTS has been activated by the modem. If CTS is not yet active, RTS remains active and as long as CTS is activated within one second, the transmission occurs. After one second, if CTS is still not activated, then RTS is set inactive and the transmission is aborted.

For modems that do not supply a CTS signal at all, tie RTS to CTS and use the shortest delay possible without losing reliable operation.

Note

If an RTS Send Delay of 0 is selected, then transmission starts as soon as CTS is activated. If CTS does not go active within 1 second after RTS is raised, RTS is set inactive and the transmission is aborted.

Certain modems will drop their carrier link when RTS is lost even though the transmission has not been finished yet. The RTS Off Delay parameter specifies in 20 millisecond increments the delay between when the last serial character is sent to the modem and when RTS is deactivated. This gives the modem extra time to transmit the last character of a packet.

ASCII Communication Protocol

ML1000	Fixed	5/01	5/02	5/03	5/04	5/05
				✓	✓	✓

The SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors support user-defined ASCII protocol by configuring RS-232 (channel 0) for the User mode. In the User mode, all received data is placed in a buffer. To access the data, use the ASCII instructions in your ladder program. See chapter 11 for more information on ASCII instructions. You can also send ASCII string data to most attached devices that accept ASCII protocol.

Note *Only ASCII instructions can be used when User mode is configured. If you use a Message (MSG) instruction that references channel 0, an error occurs.*

ASCII Channel 0 Parameter Configuration

When the user mode driver is Generic ASCII for channel 0, the following parameters can be changed:

Parameter	Description
Diagnostic File	Reserved for future use.
Baud Rate	Toggles between the communication rate of 110, 300, 600, 1.2K, 2.4K, 4.8K, 9.6K, and 19.2K (additional rate of 38.4K for SLC 5/05 only). The default is 1.2K for SLC 5/03 and SLC 5/04, and 19.2K for the SLC 5/05.
Parity	Toggles between None, Odd, and Even. The default is None.
Stop Bits	Toggles between 1, 1.5, and 2. The default is 1.
Data Bits	Toggles between 7 and 8. The default is 8.
Delete Mode	Toggles between Ignore, CRT, and printer. The default is Ignore. This parameter is dependent on the Echo parameter being Enabled.
Echo	Toggles between Disabled and Enabled. The default is Disabled.
RTS Off Delay	Allows you to select the RTS off delay value in increments of 20 ms. Valid range is 0–65535 (in 20 ms increment). The default is 0.
RTS Send Delay	Allows you to select the RTS send delay value in increments of 20 ms. Valid range is 0–65535 (in 20 ms increment). The default is 0.
Control Line	Toggles between No Handshaking, Half-Duplex With Continuous Carrier, Half-Duplex Without Continuous Carrier, and Full-Duplex Modem. The default is No Handshaking.
XON/XOFF	Toggles between Disabled and Enabled. The default is Disabled.
Termination 1 Termination 2	Specify FF for no termination character.
Append 1 Append 2	Specify FF for no append character.

Using the Passthru Features

There are three types of passthru available in the SLC 5/03 and SLC 5/04 processors. Their operation and associated bits are described below.

DH+ to DH-485 Passthru – (All SLC 5/04 processors)

This type allows the SLC 5/04 to act as a bridge between a DH+ network and a DH-485 network. When bit S:34/0 is reset, communication packets coming into channel 0 (configured for DH-485) that are not intended for the SLC 5/04 processor are resent out channel 1 onto the DH+ network. Also, communication packets coming into channel 1 (DH+) that are not intended for the SLC 5/04 processor are re-sent out channel 0 onto the DH-485 network. This activity has some effect on the scan time of the SLC 5/04 processor's ladder program, but the effects are not dramatic because only one passthru packet is re-routed per scan.

DF1 to DH+ Passthru – (SLC 5/04 OS401 and above processors)

This type allows you to connect a computer to the SLC 5/04 processor's serial port (channel 0 configured for DF1 Full-Duplex) and access any node on the DH+ network, regardless of the baud rate of the DH+ network. You can also connect a modem to the serial port and dial into any node on the DH+ network. Passthru is enabled when bit S:34/5 is set.

Remote I/O Passthru (SLC 5/03 OS302, SLC 5/04 OS401, and SLC 5/05 processors)

This type allows the SLC processor system to act as a bridge between its channel 0 and/or channel 1 network(s) and the remote I/O network supported by the 1747-SN Remote I/O module. This allows personal computers on DH+, DH-485, Ethernet, or DF1 networks to upload or download applications to devices such as PanelView 550s, PanelView 900s, and DataLiners on the remote I/O network.

Considerations when DF1 to DH+ Passthru is Enabled

Keep the following information in mind when you are using DF1 to DH+ Passthru.

Going Online with an SLC 5/04 Processor using DF1 Full-Duplex

If you want to go on-line using DF1 full-duplex, make sure the destination address under the Full-Duplex Online Configuration Screen is set to the DH+ node address channel 1 of the target SLC 5/04 processor. If the destination address is not set and the SLC 5/04 processor has the DF1 to DH+ passthru feature enabled, the command packets from the programming software may go to a different SLC 5/04 processor than the intended SLC 5/04 processor.

Sending a Message using DF1 Full-Duplex *to* an SLC 5/04 Processor with DF1 to DH+ Passthru Enabled

If the receiving SLC 5/04 processor has passthru enabled, make sure the target node parameter is set to the channel 1 DH+ address of the SLC 5/04 processor.

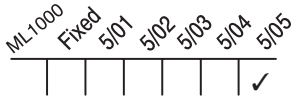
Sending a Message using DF1 Full-Duplex *from* an SLC 5/04 Processor with DF1 to DH+ Passthru Enabled

If you use an SLC 5/04 processor with DF1 to DH+ passthru enabled to send messages out of channel 0 (configured for DF1 full-duplex), you must make sure that the SLC 5/04 processor's DH+ node address appears as the DF1 source address under the Channel 0 System Mode Configuration Screen. If the address is not set correctly, responses coming back to the SLC 5/04 processor may be sent to other nodes on the DH+ network instead.

Communicating from an SLC 5/04 Processor using PLC-2® addressing

If you use an SLC 5/04 processor with DF1 to DH+ passthru enabled and are trying to send messages out of channel 0 using the MESSAGE instructions, do not use the 485 CIF message type. Use either the 500CPU or PLC5 message types. If you try to use the 485 CIF message type, the SLC 5/04 processor sending the message will not receive replies from the node it is attempting to communicate with.

Ethernet Communication Protocol



This section:

- describes SLC 5/05 performance considerations
- describes Ethernet network connections and media
- explains how the SLC 5/05 establishes node connections
- lists Ethernet configuration parameters and procedures
- describes configuration for subnet masks and gateways

The SLC 5/05 supports Ethernet communication via the Ethernet communication channel 1. Ethernet is a local area network that provides communication between various devices at 10 Mbps. The physical communication media options for the SLC 5/05 are:

- built-in
  - twisted pair (10Base-T)
- with media converters or hubs
  - fiber optic
  - broadband
  - thick-wire coaxial cable (10Base-5)
  - thin-wire coaxial cable (10Base-2)

SLC 5/05 Performance Considerations

Actual performance of an SLC 5/05 processor varies according to:

- size of Ethernet messages
- frequency of Ethernet messages
- network loading
- the implementation of and performance of your processor application program

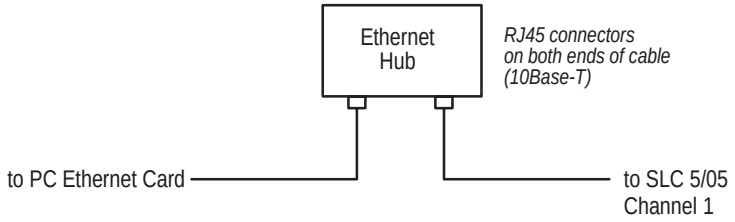
Optimal Performance: PC to SLC 5/05 Processor (2-node Ethernet network)

Operation	Words	MSG per second	ms per MSG	Words per second
Single Typed Read	1	140	281	140
Single Typed Reads	20	138	287	2760
Single Typed Reads	100	129	312	12,900

SLC 5/05 and PC Connections to the Ethernet Network

TCP/IP is the mechanism used to transport Ethernet messages. On top of TCP, the Client/Server Protocol is required to establish sessions and to send the MSG commands. Connections can be initiated by either a client program (INTERCHANGE or RSLinx application) or a processor. See page 8–23 for information on how connections are established using the MSG instruction

The SLC 5/05 Ethernet connector conforms to ISO/IEC 8802–3 STD 802.3 and utilizes 10Base-T media. Connections are made directly from the SLC 5/05 to an Ethernet hub. The network setup is simple and cost effective. Typical network topology is pictured below.



Important: *The SLC 5/05 processor contains a 10Base-T, RJ45 Ethernet connector which connects to standard Ethernet hubs via 8-wire twisted pair straight-through cable. To access other Ethernet mediums, use 10Base-T media converters or Ethernet hubs that can be connected together via fiber, thin-wire, or thick-wire coaxial cables, or any other physical media commercially available with Ethernet hubs.*

Configuring the Ethernet Channel on the SLC 5/05

There are two ways to configure the SLC 5/05 Ethernet channel 1. The configuration can be done via a BOOTP request at processor powerup, or by manually setting the configuration parameters using RSLogix 500 Programming Software. The configuration parameters are shown below and the configuration procedures follow.

Parameter	Description	Default	Status
Diagnostic File Number	The file number of the diagnostic counter for this channel. A Diagnostic File Number value of zero means that no diagnostics file was configured for this channel. The Diagnostic File Number must be an integer within the limits of 7, 9–255.	0	read/write
MSG Connection Timeout	The amount of time (in ms) allowed for a MSG instruction to establish a connection with the destination node. The MSG Connection Timeout has 250 ms resolution and a range from 250 to 65,500.	15,000 ms	read/write
MSG Reply Timeout	The amount of time (in ms) that the SLC 5/05 waits for a reply to a command that it has initiated via a MSG instruction. The MSG Reply Timeout has 250 ms resolution and a range from 250 to 65,500.	3,000 ms	read/write
Inactivity Timeout	The amount of time (in minutes) that a MSG connection may remain inactive before it is terminated. The Inactivity Timeout has a 1 minute resolution and a range from 1 to 65,500 minutes.	30 minutes	read/write
IP Address	The SLC 5/05 internet address (in network byte order). The internet address must be specified to connect to the TCP/IP network.	0 (undefined)	read/write
Subnet Mask	The SLC 5/05 subnet mask (in network byte order). The Subnet Mask is used to interpret IP addresses when the internet is divided into subnets. A Subnet Mask of all zeros indicates that no subnet mask has been configured.	0	read/write
Broadcast Address	NOT SUPPORTED AT THIS TIME. The SLC 5/05 broadcast address (in network byte order). The Broadcast Address is used in sending multicast messages. A Broadcast Address of all zeros indicates that no broadcast address had been configured. In this case, the network code chooses a valid broadcast address when needed for that current subnet.	0	
Gateway Address	The address of a gateway (in network byte order) that provides connection to another IP network. A Gateway Address of all zeros indicates that no gateway has been configured.	0	read/write
BOOTP Enable	The BOOTP enable switch. When BOOTP is enabled, the SLC 5/05 attempts to learn its network related parameters at powerup via a BOOTP request. There must be a BOOTP server on the network capable of responding to this BOOTP request. When BOOTP is disabled, the SLC 5/05 uses the locally configured network related parameters (IP Address, Subnet Mask, Broadcast Address, etc.).	1 (enabled)	read/write
Hardware Address	The SLC 5/05 Ethernet hardware address.	Ethernet hardware address	read only

Configuration Using RSLogix500 Programming Software

Refer to the documentation provided with your programming software.

Configuration Via BOOTP

BOOTP is a standard protocol that TCP/IP nodes use to obtain start-up information. By default, the SLC 5/05 broadcasts BOOTP requests at powerup. The BOOTP Valid parameter remains clear until a BOOTP reply has been received. BOOTP lets you dynamically assign IP Addresses to processors on the EthernetLink.

To use BOOTP, a BOOTP Server must exist on the local Ethernet subnet. The server is a computer that has BOOTP Server software installed and reads a text file containing network information for individual nodes on the network.

The BOOTP request can be disabled by clearing the BOOTP Enable parameter in the channel Configuration File. When BOOTP Enable is cleared (disabled), the SLC 5/05 uses the existing channel configuration data.

Important: *If BOOTP is disabled, or no BOOTP server exists on the network, you must use SLC 500 programming software to enter/change the IP address for each processor.*

The host system's BOOTP configuration file must be updated to service requests from SLC 5/05 processors. The following parameters must be configured:

Parameter	Description
IP Address	A unique IP Address for the SLC 5/05 processor.
Subnet Mask	Specifies the net and local subnet mask as per the standard on subnetting *RFC 950, Internet Standard Subnetting Procedure*.
Gateway	Specifies the IP address of a gateway on the same subnet as the SLC 5/05 that provides connections to another IP network.

Note: *If you do not have BOOTP Server capabilities on your network, and you want to dynamically configure Channel 1, contact your local Allen-Bradley representative to obtain a free BOOTP Utility diskette.*

BOOTP Operation at Power-Up

When BOOTP is enabled, the following events occur at power-up:

- The processor broadcasts a BOOTP-request message containing its hardware address over the local network or subnet.
- The BOOTP server compares the hardware address with the addresses in its look-up table in the BOOTPTAB file.
- The BOOTP server sends a message back to the processor with the IP address and other network information that corresponds to the hardware address it received.

With all hardware and IP addresses in one location, you can easily change IP addresses in the BOOTP configuration file if your network needs change.

Using DOS/Windows BOOTP

The optional BOOTP Server diskette contains DOS-based and Windows-based BOOTP server utilities. Both provide BOOTP services for SLC 5/05 processors. Regardless of the platform you are using, you must:

- install the boot-server utility
- edit the boot-server configuration file
- run the boot-server utility

Important: *Do not use the BOOTP utility disk if you already have INTERCHANGE software installed. Instead, use the boot-server capabilities that came with your INTERCHANGE software.*

Install the DOS/Windows BOOTP server

To install the DOS BOOTP server:

1. Put the utility disk that came with your processor in your disk drive.
2. Change directory to the disk drive.
3. Type **`install`**, and press **`[Enter]`**.
4. The software is installed in C:\ABIC\BIN. Put this directory in the path statement of your AUTOEXEC.BAT file.

Edit the DOS/Windows BOOTP Configuration File

The boot-server configuration file, BOOTPTAB, is located in the C:\ABIC\BIN directory. This file contains the information needed to boot SLC 5/05 processors.

You must edit the BOOTPTAB file, which is an ASCII text file, to include the name, IP address, and hardware address for each SLC 5/05 processor you want the server to boot. To edit this file:

1. Open the BOOTPTAB file using a text editor.

   The file contains lines that look like this:

   ```
   #Default string for each type of Ethernet client
   defaults5E: ht=1:vm=rfc1048
   ```

   These are the default parameters for SLC 5/05 processors and must always precede the client lines in the BOOTPTAB file.

   The file also contains a line that looks like this:

   ```
   plc5name: tc=defaults5E:ip=aa.bb.cc.dd:ha=0000BC1Dxxyy
   ```

Important: *Use this line as the configuration template for SLC 5/05 processors.*

2. Make one copy of the SLC 5/05 processor template for every SLC 5/05 processor in your system.

3. Edit each copy of the template as follows:

   a. Replace plc5name with the name of the SLC 5/05 processor. Use only letters and numbers; do not use underscores.

   b. Replace aa.bb.cc.dd with the IP address to be assigned to the processor.

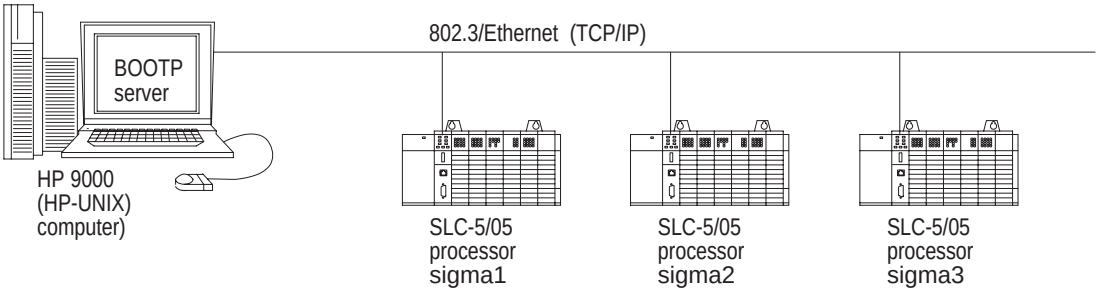
   c. Replace xxyy with the last four digits of the hardware address. Use only valid hexadecimal digits (0-9, A-F); do not use the hyphens that separate the numbers. (You will find the hardware address on a label affixed to the printed circuit board of the SLC 5/05 processor.

4. Save, close, and make a backup copy of this file.

Example

In this example there are three SLC 5/05 processors and an HP 9000 programming terminal. The names and hardware addresses are device specific:

Device	Name	IP Address	Hardware Address
SLC 5/05	sigma1	12.34.56.1	00–00–BC–1D–12–34
SLC 5/05	sigma2	12.34.56.2	00–00–BC–1D–56–78
SLC 5/05	sigma3	12.34.56.3	00–00–BC–1D–90–12



Based on this configuration, the BOOTPTAB file looks like:

```
#  Legend:   gw -- gateways
#            ha -- hardware address
#            ht -- hardware type①
#            ip -- host IP address
#            sm -- subnet mask
#            vm -- BOOTP vendor extensions format②
#            tc -- template host

#Default string for each type of Ethernet client
defaults5E: ht=1:vm=rfc1048

#Entries for SLC 5/05 processors:
sigma1: tc=defaults5E:ip=12.34.56.1:ha=0000BC1D1234
sigma2: tc=defaults5E:ip=12.34.56.2:ha=0000BC1D5678
sigma3: tc=defaults5E:ip=12.34.56.3:ha=0000BC1D9012
```

① 1 = 10MB Ethernet

② Use rfc1048

Run the Boot Server Utility

You can run either the DOS-based utility or the Windows-based BOOTP utility, but not both.

If you have BOOTP enabled and the message `BOOTP response not received` appears, check the cabling connections and the BOOTP server system.

If you're using this platform	then invoke this executable	from the
DOS-based	`DTLBOOTD.EXE`	DOS command line (specify optional parameters if necessary)
Windows	`DTLBOOTW.EXE`	Windows Program Manager

Both utilities are located in the `C:\ABIC\BIN` directory and use the information contained in the `BOOTPTAB` file.

Be sure to place the `BOOTPTAB` file in the directory from which you are running the BOOTP utility. If this file is not found in that directory, the utility will try to find the file in the directory specified by the environment variable `ABIC_CONFIG`.

Running the DOS-Based Utility

To run the boot-server utility, `DTLBOOTD.EXE`, follow these steps:

1. At the DOS prompt, type:

`DTLBOOTD` [**`-D`**] [**`-T`** *<timeout>*] [**`-B`** *<numboots>*] [**`-F`** *<numfiles>*] [*configfile*] [*logfile*]

Parameter	Description
-D	provide additional information for debug purposes.
–T <timeout>	exit after <timeout> seconds of inactivity.
–B <numboots>	exit after answering <numboots> number of boot requests.
–F <numfiles>	exit after answering <numfiles> number of file requests.
configfile	name of the boot server configuration file to use. The default configuration file is `%ABIC_CONFIG%\BOOTPTAB.`
logfile	name of the log file to use. The default log file is `%ABIC_CONFIG%\DTLBOOTD.LOG`.

Once you invoke the utility, it runs until the specified exit parameter is satisfied. Exit any time by pressing **[Esc]**.

2. Apply power to all chassis containing SLC 5/05 processors.

   At power-up, each SLC 5/05 processor broadcasts a BOOTP request if BOOTP was enabled at the channel 1 configuration screen. The Ethernet boot server compares the hardware address with those listed in BOOTPTAB and responds by sending the corresponding IP address and other configuration data to the client via a BOOTP reply.

Running the Windows-Based Utility

To run the boot-server utility, DTLBOOTW.EXE, follow these steps:

1. Start Microsoft Windows®, if it is not already running.
2. Open the Program Manager window, if it is not already open.
3. Choose File on the menu bar and select Run from the menu.
4. In the dialog box, type C:\ABIC\BIN\DTLBOOTW; then, choose OK or press [Enter].

   Once you invoke the utility, it will run until you terminate it by closing the DTLBOOTW.EXE window and exiting from Windows.

5. Apply power to all chassis containing and SLC 5/05 processors.

   At power-up, each SLC 5/05 processor broadcasts a BOOTP request. The Ethernet boot server compares the hardware address with those listed in BOOTPTAB and responds by sending the corresponding IP address and other configuration data to the client via a BOOTP reply.

Using Subnet Masks and Gateways

Configure subnet masks and gateways using the Ethernet channel 1 configuration screen:

Important: *If BOOTP is enabled, you can't change any of the advanced Ethernet communications characteristics.*

If your network is divided into subnetworks that use gateways or routers, you must indicate the following information when configuring channel 1:

- subnet mask
- gateway address

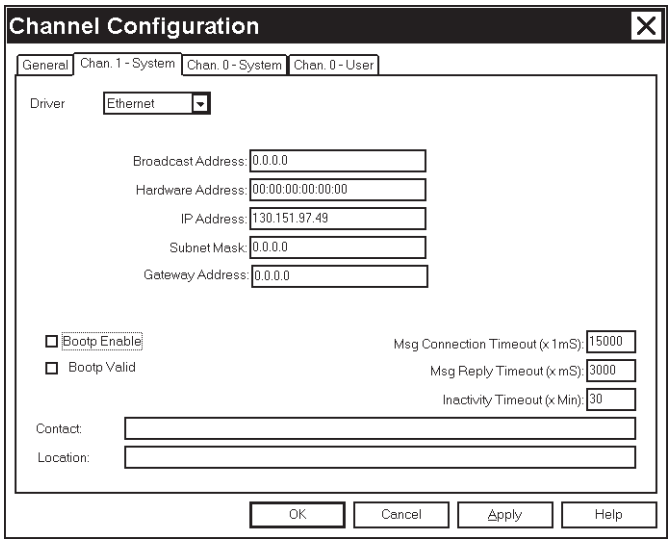
A *subnet mask* is a filter that a node applies to IP addresses to determine if an address is on the local subnet or on another subnet. If an address is located on another subnetwork, messages are routed through a local gateway to be transferred to the destination subnetwork.

If your network is not divided into subnets, then leave the subnet mask field at the default.

If you are	Then	See page
manually configuring channel 1 and have a network with subnets	• be sure the BOOTP enable field is disabled • use your programming software to enter the subnet mask and gateway address.	13–48
using BOOTP to configure channel 1 and have a network with subnets	• be sure BOOTP is enabled • configure the BOOTPTAB file to include the subnet mask(s) and gateway address(es)	13–49

Manually Configuring Channel 1 for Processors on Subnets

If you are manually configuring channel 1 for a processor located on a subnet, deselect the “BOOTP Enable” option by clicking on the checked box.



See the table below to configure the subnet mask and gateway address fields for each processor via your programming software.

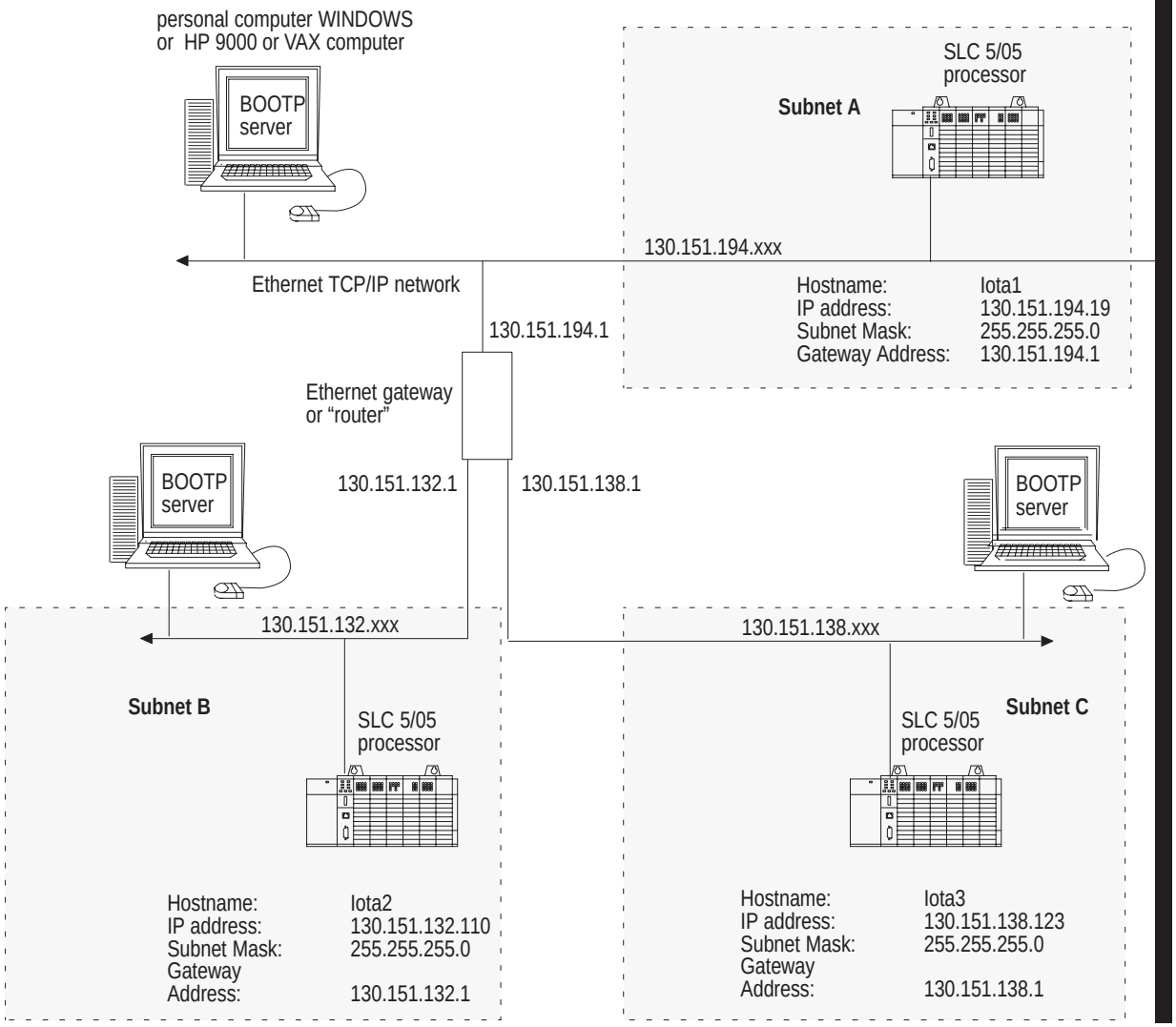
Ethernet Channel 1 Configuration Screen Advanced Functions

This field:	Specifies:	Configure by doing the following:
`Subnet Mask`	The processor’s subnet mask. The subnet mask is used to interpret IP addresses when the internet is divided into subnets.	Enter an address of the following form: a.b.c.d Where: a, b, c, d are between 0-255 (decimal) If your network is not divided into subnets, then leave the subnet mask field at the default. If you change the default and need to reset it, type 0.0.0.0.
`Gateway Address`	The IP address of the gateway that provides a connection to another IP network. This field is required when you communicate with other devices not on a local subnet.	Enter an address of the following form: a.b.c.d Where: a, b, c, d are between 0-255 (decimal) The default address is `No Gateway`.

Using BOOTP to Configure Channel 1 for Processors on Subnets

Configure the BOOTPTAB file according to the subnet mask and gateway address for each SLC 5/05 processor on the link. See the example below and the corresponding BOOTPTAB file on the next page.

Important: *Because BOOTP requests are seen only on the local subnet, each subnet needs its own BOOTP server and BOOTPTAB file.*



The BOOTPTAB files that correspond to the example looks like:

```
#  Legend:  gw -- gateways
#           ha -- hardware address
#           ht -- hardware type
#           ip -- host IP address
#           sm -- subnet mask
#           vm -- BOOTP vendor extensions format
#           tc -- template host
#Default string for each type of Ethernet client
defaults5E: ht=1:vm=rfc1048:sm=255.255.255.0
#Entries for SLC 5/05 processors:
iota1:\
       tc=defaults5E:\
       gw=130.151.194.1:\
       ha=0000BC1D1234:/
       ip=130.151.194.19
```

```
#  Legend:  gw -- gateways
#           ha -- hardware address
#           ht -- hardware type
#           ip -- host IP address
#           sm -- subnet mask
#           vm -- BOOTP vendor extensions format
#           tc -- template host
#Default string for each type of Ethernet client
defaults5E: ht=1:vm=rfc1048:sm=255.255.255.0
#Entries for SLC 5/05 processors:
iota2:\
       tc=defaults5E:\
       gw=130.151.132.1:\
       ha=0000BC1D5678:/
       ip=130.151.132.110
```

```
#  Legend:  gw -- gateways
#           ha -- hardware address
#           ht -- hardware type
#           ip -- host IP address
#           sm -- subnet mask
#           vm -- BOOTP vendor extensions format
#           tc -- template host
#Default string for each type of Ethernet client
defaults5E: ht=1:vm=rfc1048:sm=255.255.255.0
#Entries for SLC 5/05 processors:
iota3:\
       tc=defaults5E:\
       gw=130.151.138.1:\
       ha=0000BC1D9012:/
       ip=130.151.138.123
```

14 MicroLogix Communication Protocols

Use the information in this chapter to understand the differences in communication protocols. The following protocols are supported:

- DF1 Full-Duplex and DF1 Half-Duplex Slave

  All MicroLogix 1000 controllers support the DF1 protocol from the RS-232 connector.

- DH-485

  Series C or later MicroLogix 1000 controllers can communicate on DH-485 networks using an AIC+ Advanced Interface Converter.

This chapter starts out with information on automatic protocol switching. This feature allows you easily switch between DF1 and DH-485 protocols. For information about required network connecting equipment, refer to the *MicroLogix 1000 Programmable Controllers User Manual*, publication 1761-6.3.

Automatic Protocol Switching

The Series D and MicroLogix 1000 analog controllers perform automatic protocol switching between DH-485 and the configured DF1 protocol. (The controller cannot automatically switch between DF1 full-duplex and DF1 half-duplex slave.) This feature allows you to switch from active communication on a DF1 half-duplex network to the DH-485 protocol to make program changes.

If the controller is configured for DF1 half-duplex slave and you'd like to switch to DH-485 protocol to edit your program, simply disconnect the MicroLogix controller from the half-duplex modem and connect it to your personal computer. The controller recognizes the computer is attempting to communicate using the DH-485 protocol and automatically switches to it. When your program changes are complete, you can disconnect your computer, reconnect the modem, and the controller automatically switches back to DF1 half-duplex slave protocol.

The following baud rate limitations affect autoswitching:

- If the configured DH-485 baud rate is 19200, the configured DF1 baud rate must be 4800 or greater.
- If the configured DH-485 baud rate is 9600, the configured DF1 baud rate must be 2400 or greater.

RS-232 Communication Interface

RS-232 is an Electronics Industries Association (EIA) standard that specifies the electrical, mechanical, and functional characteristics for serial binary communication. It provides you with a variety of system configuration possibilities. (RS-232 is a definition of electrical characteristics; it is *not* a protocol.)

One of the biggest benefits of the RS-232 interface is that it lets you integrate telephone and radio modems into your control system. The distance over which you are able to communicate with certain system devices is virtually limitless.

DF1 Full-Duplex Protocol

DF1 Full-Duplex communication protocol (also referred to as DF1 point-to-point protocol) combines data transparency (ANSI — American National Standards Institute — specification subcategory D1) and 2-way simultaneous transmission with embedded responses (subcategory F1).

The MicroLogix 1000 controllers support the DF1 Full-Duplex protocol via RS-232 connection to external devices, such as computers, the Hand-Held Programmer (catalog number 1761-HHP-B30), or other MicroLogix 1000 controllers. (For information on connecting to the Hand-Held Programmer, see its user manual, publication 1761-6.2.)

Note

You can connect to a DeviceNet network using the DeviceNet Interface (DNI), catalog number 1761-NET-DNI. The DNI provides a single DeviceNet connection point and a single RS-232 connection. The DNI is a DeviceNet to DF1 protocol conversion device that allows DF1 devices to communicate on DeviceNet.

For additional information on using the DNI, see the DeviceNet Interface (DNI) Overview, publication 1761-1.23, or the DeviceNet Interface (DNI) User Manual, publication 1761-6.5.

DF1 Full-Duplex Operation

DF1 Full-Duplex protocol (also referred to as DF1 point-to-point protocol) is useful where RS-232 point-to-point communication is required. This type of protocol supports simultaneous transmissions between two devices in both directions. DF1 protocol controls message flow, detects and signals errors, and retries if errors are detected.

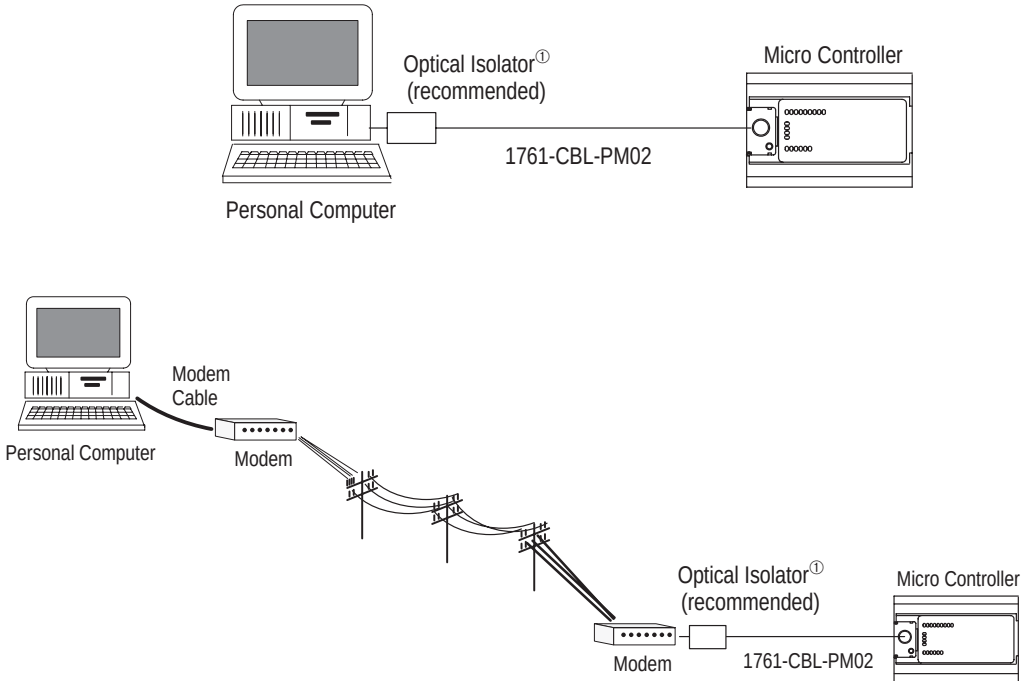
DF1 Full-Duplex Channel 0 Configuration Parameters

When the system mode driver is a DF1 Full-Duplex for channel 0 the following parameters can be changed:

Parameter	Description
Diagnostic File	Reserved for future use.
Baud Rate	Toggles between the communication rate of 110, 300, 600, 1200, 2400, 4800, 9600, and 19.2K. (additional rate of 38.4K for SLC 5/05 only). The default is 1200 for SLC 5/05 and SLC 5/04, and 19.2K for the SLC 5/05.
Parity	Toggle between None and Even. The default is None.
Stop Bits	Toggles between 1, 1.5, and 2. The default is 1.
Duplicate Packet Detection	Toggles between Disabled and Enabled. The default is Enabled.
Error Detection	Toggles between CRC and BCC. The default is CRC.
ACK Timeout	Valid range is 2–65535 (in 20 ms increment) The default is 50.
NAK Retries	Valid range is 0–255. The default is 3.
ENQ Retries	Valid range is 0–255. The default is 3.
Control Line	Toggles between No Handshaking and Full-Duplex modem. The default is No Handshaking.
Embedded Responses	Toggles between Enabled and Auto-Detect. The default is Enabled.
Source ID	Specify the address of the sender in this field. Valid range is 0–254. The default is 9.

Example DF1 Full-Duplex Connections

For information about required network connecting equipment, refer to the *MicroLogix 1000 Programmable Controllers User Manual*, publication 1761-6.3.



① We recommend using an AIC+, catalog number 1761-NET-AIC, as your optical isolator.

DF1 Half-Duplex Slave Protocol

DF1 half-duplex slave protocol provides a multi-drop single master/multiple slave network. In contrast to DF1 full-duplex, communication takes place in one direction at a time. You can use the RS-232 port on the MicroLogix as both a half-duplex programming port, as well as a half-duplex peer-to-peer messaging port.

The master device initiates all communication by “polling” each slave device. The slave device may only transmit message packets when it is polled by the master. It is the master’s responsibility to poll each slave on a regular and sequential basis to allow slaves to send message packets back to the master. During a polling sequence, the master polls a slave either repeatedly until the slave indicates that it has no more message packets to transmit or just one time per polling sequence, depending on how the master is configured.

An additional feature of the DF1 half-duplex protocol is that it is possible for a slave device to enable a MSG instruction in its ladder program to send or request data to/from another slave. When the initiating slave is polled, the MSG instruction command packet is sent to the master. The master recognizes that the command packet is not intended for it but for another slave, so the master immediately rebroadcasts the command packet to the intended slave. When the intended slave is polled, it sends a reply packet to the master with the data the first slave requested. The master again recognizes that the reply packet is intended for another slave, so the master immediately rebroadcasts the reply packet to that slave. This slave-to-slave transfer is a function of the master device and is also used by programming software to upload and download programs to processors on the DF1 half-duplex link.

Several Allen-Bradley products support half-duplex master protocol. They include the SLC 5/03™ and 5/04™, 1771-KGM module (for PLC-2® controllers) and enhanced PLC-5® processors. Rockwell Software WINtelligent LINX™ also supports DF1 half-duplex master protocol.

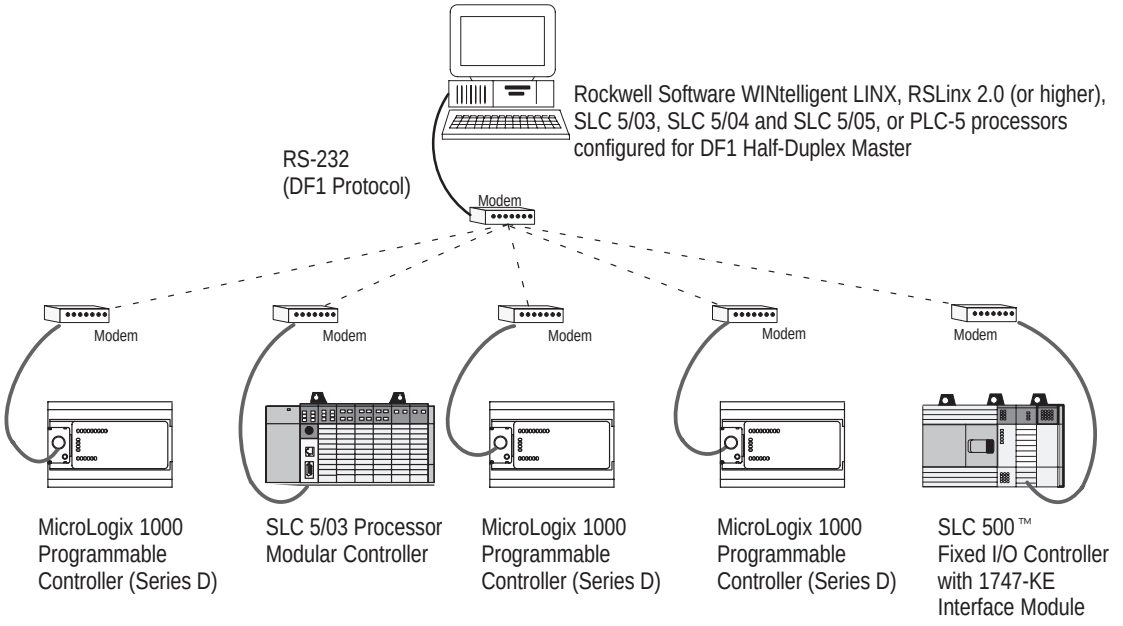
Typically, the master maintains an active node table that indicates which slaves are active (slaves that responded the last time they were polled) and which slaves are inactive (slaves that did not respond the last time they were polled). The active slaves are polled on a regular basis. The inactive slaves are only polled occasionally to check if any have come back online.

DF1 half-duplex supports up to 255 devices (address 0 to 254) with address 255 reserved for master broadcasts. The MicroLogix supports broadcast reception but cannot initiate a broadcast command. The MicroLogix supports half-duplex modems using Request-To-Send/Clear-To-Send (RTS/CTS) hardware handshaking.

DF1 Half-Duplex Slave Configuration Parameters

When the system mode driver is DF1 half-duplex slave the following parameters can be viewed and changed only when the programming software is online with the processor. The DF1 half-duplex slave parameters are not stored as part of the controller downloadable image *(with the exception of the baud rate and node address)*. If a failed MicroLogix 1000 controller is replaced and the backed-up controller image is downloaded to the replacement controller, these parameters will remain at default until manually changed. Therefore, be sure to fully document any non-default settings to the DF1 half-duplex slave configuration parameters.

Parameter	Description	Default
Baud Rate	Toggles between the communication rate of 300, 600, 1200, 2400, 4800, 9600, 19.2K (additional rate of 38.4K for SLC 5/05 only). The default is 1200 for SLC 5/03 and SLC 5/04, and 19.2K for the SLC 5/05.	9600
Node Address	Valid range is 0–254 decimal.	1
Control Line	Toggles between No Handshaking and Half-Duplex Modem.	No Hand-shaking
Duplicate Packet Detection	Detects and eliminates duplicate responses to a message. Duplicate packets may be sent under “noisy” communication conditions when the sender’s retries are not set to 0. Toggles between Enabled and Disabled.	Enabled
Error Detection	Toggles between CRC and BCC.	CRC
RTS Off Delay	Specifies the delay time between when the last serial character is sent to the modem and when RTS will be deactivated. Gives modem extra time to transmit the last character of a packet. The valid range is 0–255 and can be set in increments of 5 ms.	0
RTS Send Delay	Specifies the time delay between setting RTS (request to send) until checking for the CTS (clear to send) response. For use with modems that are not ready to respond with CTS immediately upon receipt of RTS. The valid range is 0–255 and can be set in increments of 5 ms.	0
Poll Timeout	Poll Timeout only applies when a slave device initiates a MSG instruction. It is the amount of time that the slave device waits for a poll from the master device. If the slave device does not receive a poll within the Poll Timeout, a MSG instruction error is generated, and the ladder program needs to requeue the MSG instruction. The valid range is 0–65535 and can be set in increments of 20 ms. If you are using a MSG instruction, it is recommended that a Poll Timeout value of zero not be used. Poll Timeout is disabled if set to zero.	3000 (60s)
Pre-send Time Delay	Delay time before transmission. Required for 1761-NET-AIC physical half-duplex networks. The 1761-NET-AIC needs delay time to change from transmit to receive mode. The valid range is 0–255 and can be set in increments of 5 ms.	0
Message Retries	Specifies the number of times a slave device attempts to resend a message packet when it does not receive an ACK from the master device. For use in noisy environments where message packets may become corrupted in transmission. The valid range is 0–255.	3
EOT Suppression	Slave does not respond when polled if no message is queued. Saves modem transmission power when there is no message to transmit. Toggles between Yes and No.	No



Considerations When Communicating as a DF1 Slave on a Multi-drop Link

When communication is between either your programming software and a MicroLogix 1000 Programmable Controller or between two MicroLogix Programmable Controllers via a slave-to-slave connection on a larger multi-drop link, the devices depend on a DF1 Master to give each of them polling permission to transmit in a timely manner. As the number of slaves increases on the link (up to 254), the time between when your programming software or the MicroLogix Controller is polled also increases. This increase in time may become larger if you are using low baud rates.

As these time periods grow, the following values may need to be changed to avoid loss of communication:

- programming software - increase poll timeout and reply timeout values
- MicroLogix Programmable Controller - increase poll timeout

Ownership Timeout

When a program download sequence is started by a software package to download a ladder logic program to a MicroLogix controller, the software takes “file ownership” of the processor. File ownership prevents other devices from reading from or writing to the processor while the download is in process. If the controller were to respond to a device’s read commands during the download, the processor could respond with incorrect information. Similarly, if the controller were to accept information from other devices, the information could be lost because the program download sequence could immediately overwrite the information. Once the download is completed, the programming software returns the file ownership to the controller, so other devices can communicate with it again.

With the addition of DF1 half-duplex slave protocol, the controller clears the file ownership if no supported commands are received from the owner within the timeout period. If the file ownership were not cleared after a download sequence interruption, the processor would not accept commands from any other devices because it would assume another device still had file ownership.

If a download sequence is interrupted, due to noise caused by electromagnetic interference, discontinue communications to the controller for the *ownership timeout* period and restart the program download. The *ownership timeout* period is set to 60 seconds as a default for all protocols. However, if you are using DF1 half-duplex, and the *poll timeout* value is set to greater than 60 seconds, the *poll timeout* value will be used instead of the *ownership timeout*. After the timeout, you can re-establish communications with the processor and try the program download again. The only other way to clear file ownership is to cycle power on the processor.

Using Modems that Support DF1 Communication Protocols

The types of modems that you can use with MicroLogix 1000 controllers include dial-up phone modems, leased-line modems, radio modems and line drivers. For point-to-point full-duplex modem connections that do not require any modem handshaking signals to operate, use DF1 full-duplex protocol. For point-to-multipoint modem connections, or for point-to-point modem connections that require Request-to-Send/Clear-To-Send (RTS/CTS) handshaking, use DF1 half-duplex slave protocol. In this case, one (and only one) of the other devices must be configured for DF1 half-duplex master protocol. Do not attempt to use DH-485 protocol through modems under any circumstance.

Note *Only Series D or later MicroLogix 1000 discrete controllers and all MicroLogix 1000 analog controllers support RTS/CTS modem handshaking and only when configured for DF1 half-duplex slave protocol with the control line parameter set to "Half-Duplex Modem". No other modem handshaking lines (i.e. Data Set Ready, Carrier Detect and Data Terminal Ready) are supported by any MicroLogix 1000 controllers.*

Dial-Up Phone Modems

Dial-up phone line modems support point-to-point full-duplex communications. Normally a MicroLogix 1000 controller, on the receiving end of the dial-up connection, will be configured for DF1 full-duplex protocol. The modem connected to the MicroLogix 1000 controller must support auto-answer and must not require any modem handshaking signals from the MicroLogix 1000 (i.e., DTR or RTS) in order to operate. The MicroLogix 1000 has no means to cause its modem to initiate or disconnect a phone call, so this must be done from the site of the remote modem.

Leased-Line Modems

Leased-line modems are used with dedicated phone lines that are typically leased from the local phone company. The dedicated lines may be in a point-to-point topology supporting full-duplex communications between two modems or in a point-to-multipoint topology supporting half-duplex communications between three or more modems. In the point-to-point topology, configure the MicroLogix 1000 controllers for DF1 full-duplex protocol (as long as the modems used do not require DTR or RTS to be high in order to operate). In the point-to-multipoint topology, configure the MicroLogix 1000 controllers for DF1 half-duplex slave protocol with the control line parameter set to "Half-Duplex Modem".

Radio Modems

Radio modems may be implemented in a point-to-point topology supporting either half-duplex or full-duplex communications, or in a point-to-multipoint topology supporting half-duplex communications between three or more modems. In the point-to-point topology using full-duplex radio modems, configure the MicroLogix 1000 controllers for DF1 full-duplex protocol (as long as the modems used do not require DTR or RTS to be high in order to operate). In the point-to-point topology using half-duplex radio modems, or point-to-multipoint topology using half-duplex radio modems, configure the MicroLogix 1000 controllers for DF1 half-duplex slave protocol. If these radio modems require RTS/CTS handshaking, configure the control line parameter to “Half-Duplex Modem”.

Line Drivers

Line drivers, also called short-haul “modems”, do not actually modulate the serial data, but rather condition the electrical signals to operate reliably over long transmission distances (up to several miles). Allen-Bradley’s AIC+ Advanced Interface Converter is a line driver that converts an RS-232 electrical signal into an RS-485 electrical signal, increasing the signal trasmission distance from 50 to 4000 feet. In a point-to-point line driver topology, configure the MicroLogix 1000 controller for DF1 full-duplex protocol (as long as the line drivers do not require DTR or RTS to be high in order to operate). In a point-to-multipoint line driver topology, configure the MicroLgoix 1000 controllers for DF1 half-duplex slave protocol. If these line drivers require RTS/CTS handshaking, configure the control line parameter to “Half-Duplex Modem”.

DH-485 Communication Protocol

The information in this section describes the DH-485 network functions, network architecture, and performance characteristics. It will also help you plan and operate the MicroLogix 1000 in a DH-485 network.

Note *Only Series C or later MicroLogix 1000 controllers support the DH-485 network.*

DH-485 Network Description

The DH-485 protocol defines the communication between multiple devices that co-exist on a single pair of wires. This protocol uses RS-485 half-duplex as its physical interface. (RS-485 is a definition of electrical characteristics; it is *not* a protocol.) RS-485 uses devices that are capable of co-existing on a common data circuit, thus allowing data to be easily shared between devices.

The DH-485 network offers:

- interconnection of 32 devices
- multi-master capability
- token passing access control
- the ability to add or remove nodes without disrupting the network
- maximum network length of 1219 m (4000 ft)

The DH-485 protocol supports two classes of devices: initiators and responders. All initiators on the network get a chance to initiate message transfers. To determine which initiator has the right to transmit, a token passing algorithm is used.

The following section describes the protocol used to control message transfers on the DH-485 network.

DH-485 Token Rotation

A node holding the token can send any valid packet onto the network. Each node is allowed only one transmission (plus two retries) each time it receives the token. After a node sends one message packet, it attempts to give the token to its successor by sending a “token pass” packet to its successor.

If no network activity occurs, the initiator sends the token pass packet again. After two retries (a total of three tries) the initiator will attempt to find a new successor.

The allowable range of the node address of an initiator is 0 to 31. The allowable address range for all responders is 1 to 31. There must be at least one initiator on the network.

DH-485 Configuration Parameters

When the system mode driver for channel 0 or channel 1 is DH-485 Master, the following parameters can be changed:

Parameter	Description
Diagnostic File	Reserved for future use.
Baud Rate	Toggles between the communication rate of 110, 300, 600, 1200, 2400, 9600, and 19.2K. The default is 19.2K.
Node Address	This is the node address of the processor on the DH-485 network. The valid range is 1–31. The default is 1.
Max Node Address	This is the maximum node address of an active processor. The valid range is 1–31. The default is 31.
Token Hold Factor	Determines the number of transactions allowed to make each DH-485 token rotation. Increasing this value allows your processor to increase its DH-485 throughput. This also decreases throughput to other processors on the DH-485 link. The valid range is 1–4. The default is 1. The SLC 5/01 and SLC 5/02 processors are factory set to 1.

DH-485 Network Initialization

Network initialization begins when a period of inactivity exceeding the time of a link dead timeout is detected by an initiator on the network. When the time for a link dead timeout is exceeded, usually the initiator with the lowest address claims the token. When an initiator has the token it will begin to build the network. The network requires at least one initiator to initialize it.

Building a network begins when the initiator that claimed the token tries to pass the token to the successor node. If the attempt to pass the token fails, or if the initiator has no established successor (for example, when it powers up), it begins a linear search for a successor starting with the node above it in the addressing.

When the initiator finds another active initiator, it passes the token to that node, which repeats the process until the token is passed all the way around the network to the first node. At this point, the network is in a state of normal operation.

Devices that use the DH-485 Network

In addition to the Series C or later MicroLogix 1000 controllers, the devices shown in the following table also support the DH-485 network.

Note *You cannot connect the Hand-Held Programmer, 1761-HHP-B30, to the AIC+.*

Catalog Number	Description	Installation Requirement	Function	Publication
1747-L511, -L514, -L524, -L531, -L532 -L541, -L542, -L543, -L551, -L552, -L553	SLC 500 Processors	SLC Chassis	These processors support a variety of I/O requirements and functionality.	1747-6.2
1746-BAS	BASIC Module	SLC Chassis	Provides an interface for SLC 500 devices to foreign devices. Program in BASIC to interface the 3 channels (2 RS232 and 1 DH-485) to printers, modems, or the DH-485 network for data collection.	1746-6.1 1746-6.2 1746-6.3
1785-KA5	DH+™ /DH-48 5 Gateway	(1771) PLC Chassis	Provides communication between stations on the PLC-5® (DH+) and SLC 500 (DH-485) networks. Enables communication and data transfer from PLC® to SLC 500 on DH-485 network. Also enables programming software programming or data acquisition across DH+ to DH-485.	1785-6.5.5 1785-1.21
2760-RB	Flexible Interface Module	(1771) PLC Chassis	Provides an interface for SLC 500 (using protocol cartridge 2760-SFC3) to other A-B PLCs and devices. Three configurable channels are available to interface with Bar Code, Vision, RF, Dataliner™, and PLC systems.	2760-ND001
1784-KTX, -KTXD	PC DH-485 IM	IBM XT/AT Computer Bus	Provides DH-485 using RSLinx	1784-6.5.22
1784-PCMK	PCMCIA IM	PC and Interchange	Provides DH-485 using RSLinx	1784-6.5.19
1747-PT1	Hand-Held Terminal	NA	Provides hand-held programming, monitoring, configuring, and troubleshooting capabilities for SLC 500 processors.	1747-NP002
1747-DTAM, 2707-L8P1, -L8P2, -L40P1, -L40P2, -V40P1, -V40P2, -V40P2N, -M232P3, -M485P3	DTAM, DTAM Plus, DTAM Micro	Panel Mount	Provides electronic operator interface for SLC 500 processors.	1747-ND013 2707-800, 2707-803
2711-K5A2, -B5A2, -K5A5, -B5A5, -K5A1, -B5A1, -K9A2, -T9A2, -K9A5, -T9A5, -K9A1, -T9A1	PanelView 550 and 900	Panel Mount	Provides electronic operator interface for SLC 500 processors.	2711-802, 2711-816

NA = Not Applicable

Important DH-485 Network Planning Considerations

Carefully plan your network configuration before installing any hardware. Listed below are some of the factors that can affect system performance:

- amount of electrical noise, temperature, and humidity in the network environment
- number of devices on the network
- connection and grounding quality in installation
- amount of communication traffic on the network
- type of process being controlled
- network configuration

The major hardware and software issues you need to resolve before installing a network are discussed in the following sections.

Hardware Considerations

You need to decide the length of the communication cable, where you route it, and how to protect it from the environment where it will be installed.

When the communication cable is installed, you need to know how many devices are to be connected during installation and how many devices will be added in the future. The following sections will help you understand and plan the network.

Number of Devices and Length of Communication Cable

You must install an AIC+ Advanced Interface Converter, catalog number 1761-NET-AIC, for each node on the network. If you plan to add nodes later, provide additional advanced interface converters during the initial installation to avoid recabling after the network is in operation.

The maximum length of the communication cable is 1219 m (4000 ft). This is the total cable distance from the first node to the last node on the network.

Planning Cable Routes

Follow these guidelines to help protect the communication cable from electrical interference:

- Keep the communication cable at least 1.52 m (5 ft) from any electric motors, transformers, rectifiers, generators, arc welders, induction furnaces, or sources of microwave radiation.
- If you must run the cable across power feed lines, run the cable at right angles to the lines.
- If you do not run the cable through a contiguous metallic wireway or conduit, keep the communication cable at least 0.15 m (6 in.) from ac power lines of less than 20A, 0.30 m (1 ft) from lines greater than 20A, but only up to 100k VA, and 0.60 m (2 ft) from lines of 100k VA or more.
- If you run the cable through a contiguous metallic wireway or conduit, keep the communication cable at least 0.08 m (3 in.) from ac power lines of less than 20A, 0.15 m (6 in.) from lines greater than 20A, but only up to 100k VA, and 0.30 m (1 ft) from lines of 100k VA or more.

  Running the communication cable through conduit provides extra protection from physical damage and electrical interference. If you route the cable through conduit, follow these additional recommendations:

  - Use ferromagnetic conduit near critical sources of electrical interference. You can use aluminum conduit in non-critical areas.
  - Use plastic connectors to couple between aluminum and ferromagnetic conduit. Make an electrical connection around the plastic connector (use pipe clamps and the heavy gauge wire or wire braid) to hold both sections at the same potential.
  - Ground the entire length of conduit by attaching it to the building earth ground.
  - Do not let the conduit touch the plug on the cable.
  - Arrange the cables loosely within the conduit. The conduit should contain only serial communication cables.
  - Install the conduit so that it meets all applicable codes and environmental specifications.

For more information on planning cable routes, see *Industrial Automation Wiring and Grounding Guidelines*, Publication Number 1770-4.1.

Software Considerations

Software considerations include the configuration of the network and the parameters that can be set to the specific requirements of the network. The following are major configuration factors that have a significant effect on network performance:

- number of nodes on the network
- addresses of those nodes
- baud rate

The following sections explain network considerations and describe ways to select parameters for optimum network performance (speed). See your programming software's user manual for more information.

Number of Nodes

The number of nodes on the network directly affects the data transfer time between nodes. Unnecessary nodes (such as a second programming terminal that is not being used) slow the data transfer rate. The maximum number of nodes on the network is 32.

Setting Node Addresses

The best network performance occurs when node addresses are assigned in sequential order. Initiators, such as personal computers, should be assigned the lowest numbered addresses to minimize the time required to initialize the network. The valid range for the MicroLogix 1000 controllers is 1–31 (controllers cannot be node 0). The default setting is 1. The node address is stored in the controller status file (S:16L).

If some nodes are connected on a temporary basis, do not assign addresses to them. Simply create nodes as needed and delete them when they are no longer required.

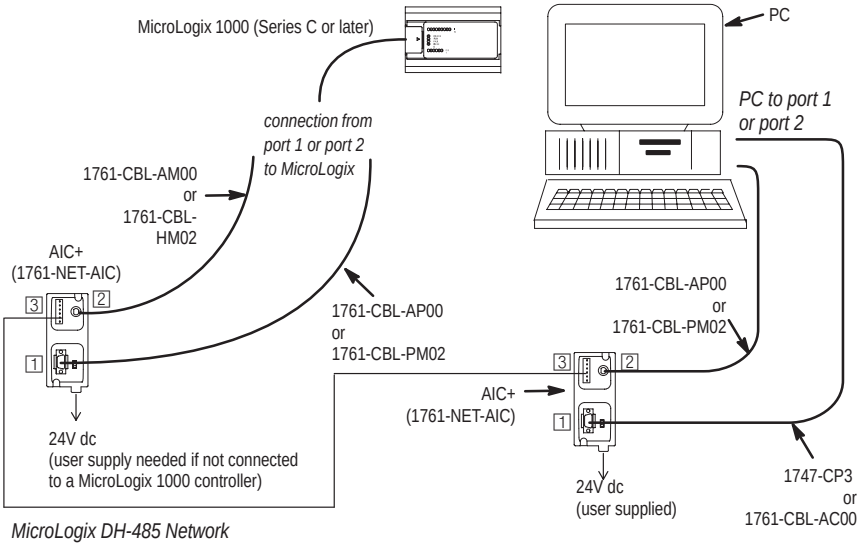
Setting Controller Baud Rate

The best network performance occurs at the highest baud rate, which is 19.2K. This is the default baud rate for a MicroLogix 1000 devices on the DH-484 network. All devices must be at the same baud rate. This rate is stored in the controller status file (S:16H).

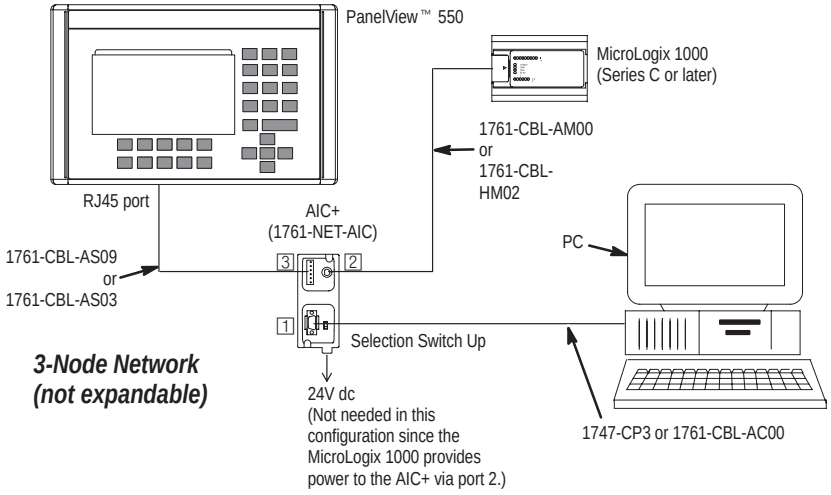
Example DH-485 Connections

The following network diagrams provide examples of how to connect Series C or later MicroLogix 1000 controllers to the DH-485 network using the AIC+.

DH-485 Network with a MicroLogix 1000 Controller

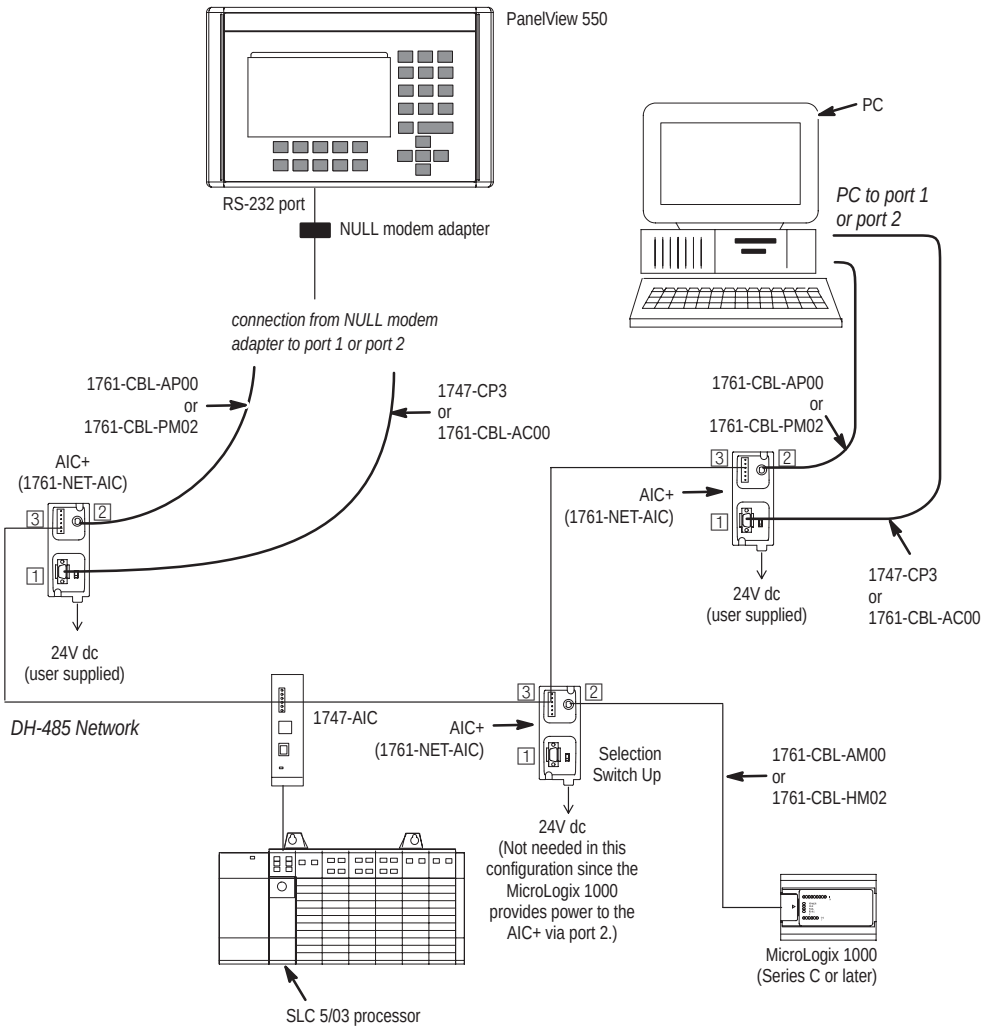


Typical 3-Node Network



1 DB-9 RS-232 port
2 mini-DIN 8 RS-232 port
3 DH-485/DF1 port

Networked Operator Interface Device and MicroLogix Controller



1 DB-9 RS-232 port

2 mini-DIN 8 RS-232 port

3 DH-485/DF1 port

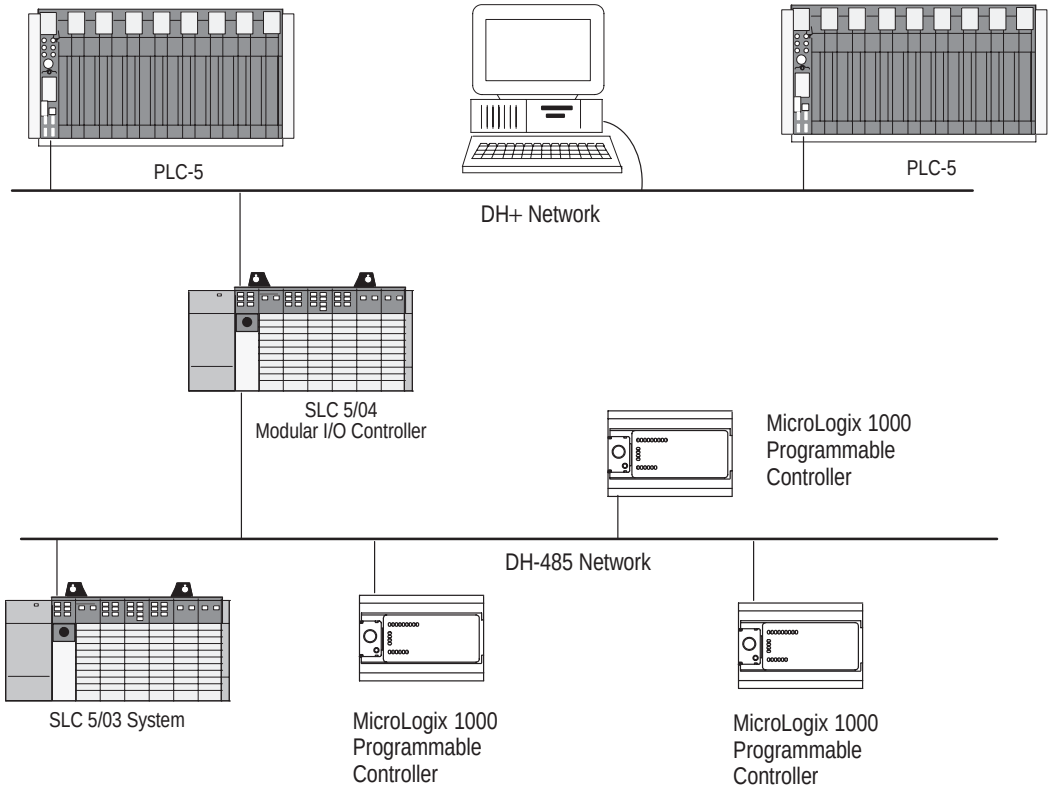
MicroLogix Remote Packet Support

Series D MicroLogix can respond to communication packets (or commands) that do not originate on the local DH-485 network. This is useful in installations that utilize different Allen-Bradley communication networks, such as DH+ and DH-485.

The example below shows how to send messages from a PLC device or a PC on the DH+ network to a MicroLogix 1000 controller on the DH-485 network. This method uses an SLC 5/04 processor bridge connection.

When using this method:

- PLC-5 devices can send read and write commands to MicroLogix controllers.
- MicroLogix 1000 controllers can respond to MSG instructions received. The MicroLogix controllers cannot initiate MSG instructions to devices on the DH+ network.
- PC can send read and write commands to MicroLogix controllers.
- PC can do remote programming of MicroLogix controllers.



15 Troubleshooting Faults

This chapter lists the major error fault codes, indicates the probable causes of faults, and recommends corrective action. This chapter also explains the operating system download faults for the SLC 5/03 (and higher) processors and MicroLogix 1000 controllers.

Automatically Clearing Faults

The following section describes the different ways to automatically clear a fault using your programming software.

SLC Processors

- Set the Fault Override at Powerup Bit S:1/8 in the status file to clear the fault when power is cycled, assuming the user program is not corrupt.
- Set one of the autoload bits S:1/10, S:1/11, or S:1/12 in the status file of the program in an EEPROM to automatically transfer a new non-faulted program from the memory module to RAM when power is cycled.

Refer to appendix B in this manual for more information on status bits S:1/13, S:1/8, S:1/10, S:1/11, S:1/12, S:5/0–7, and S:36/0–7.

Note *You can declare your own application-specific major fault by writing your own unique value to S:6 and then setting bit S:1/13.*

MicroLogix 1000 Controllers

You can automatically clear a fault when cycling power to the controller by setting either one or both of the following status bits in the status file:

- Fault Override at Powerup bit (S:1/8)
- Run Always bit (S:1/12)

Clearing a fault using the Run Always bit (S:1/12) causes the controller to immediately enter the REM Run mode. Make sure you fully understand the use of this bit before incorporating it into your program. Refer to page A–6 for more information. Also refer to chapter 1 for information pertaining to retentive data.

Note *You can declare your own application-specific major fault by writing your own unique value to S:6 and then setting bit S:1/13 to prevent reusing system defined codes. The recommended values for user defined faults is FF00 to FF0F.*

Manually Clearing Faults (SLC Processors)

The following section describes the different ways to manually clear a fault when using an SLC processor.

- Manually clear the major fault bit S:1/13, and the minor and major error bits S:5/0–7 in the status file, using a programming device or a Data Table Access Module. Place the processor in the REM Program mode. Correct the condition causing the fault, then return the processor to either REM Run or any of the REM Test modes.
- *SLC 5/03 and higher processors* – Toggle the keyswitch from RUN to PROGram and then back to RUN.

***SLC 5/03 and higher processors* – Clearing these bits with the keyswitch in the RUN position causes the processor to immediately enter the Run mode.**

If you are online with a SLC 5/03 (or higher) processor with the keyswitch position is in RUN and you press the clear major fault function key, you are warned that the processor will enter the Run mode once you clear the fault.

Using the Fault Routine

SLC Processors

When designating a subroutine file, the occurrence of recoverable or non-recoverable user faults causes the designated subroutine to be executed for one scan. If the fault is recoverable, the subroutine can be used to correct the problem and clear the fault bit S:1/13. The processor then continues in the run mode. If the fault is non-recoverable, the subroutine can send a message via the Message instruction to another node with error code information and/or does an orderly shutdown of the process.

The subroutine does not execute for non-user faults. The user fault routine is discussed in chapter 12.

MicoLogix 1000 Controllers

The occurrence of recoverable or non-recoverable user faults causes file 3 to be executed. If the fault is recoverable, the subroutine can be used to correct the problem and clear the fault bit S:1/13. The controller then continues in the REM Run mode. The subroutine does not execute for non-user faults.

Fault Messages

This section contains fault messages that can occur during operation for the MicroLogix 1000 controllers and the SLC processors. Each table lists the error code description, the probable cause, and the recommended corrective action.

MicroLogix 1000 Controller Faults

The controller faults are divided into the following types:

- powerup errors
- going-to-run errors
- run errors
- download errors

Powerup Errors

Error Code (Hex)	Advisory Message	Description	Recommended Action
0001	DEFAULT PROGRAM LOADED	The default program is loaded to the controller memory. This occurs: • on power-up if the power down occurred in the middle of a download • if the user program is corrupt at power-up, the default program is loaded.	• Re–download the program and enter the REM Run mode. • Contact your local Allen–Bradley representative if the error persists.
0002	UNEXPECTED RESET	The controller was unexpectedly reset due to a noisy environment or internal hardware failure. If the user program downloaded to the controller is valid, the initial data downloaded with the program is used. The Retentive Data Lost Bit (S:5/8) is set. If the user program is invalid, the default program is loaded.	• Refer to proper grounding guidelines in the *MicroLogix 1000 Programmable Controllers User Manual*, publication 1761-6.3. • Contact your local Allen–Bradley representative if the error persists.
0003	EEPROM MEMORY IS CORRUPT	The user program is corrupt and the default program is loaded.	• While power cycling to your controller, a noise problem may have occurred. Try cycling power again. Your program may be valid, but retentive data will be lost. • Contact your local Allen–Bradley representative if the error persists.
0005	RETENTIVE DATA HAS BEEN LOST	The data files (input, output, timer, counter, integer, binary, control, and status) are corrupt.	• Cycle power on your unit. • Download your program and re–initialize any necessary data. • Start up your system. • Contact your local Allen–Bradley representative if the error persists.

Going-to-Run Errors

Error Code (Hex)	Advisory Message	Description	Recommended Action
0008	FATAL INTERNAL SOFTWARE ERROR	The controller software has detected an invalid condition within the hardware or software after completing power-up processing (after the first 2 seconds of operation).	• Cycle power on your unit. • Download your program and re–initialize any necessary data. • Start up your system. • Contact your local Allen–Bradley representative if the error persists.
0009	FATAL INTERNAL HARDWARE ERROR	The controller software has detected an invalid condition within the hardware during power-up processing (within the first 2 seconds of operation).	• Cycle power on your unit. • Download your program and re–initialize any necessary data. • Start up your system. • Contact your local Allen–Bradley representative if the error persists.
0010	INCOMPATIBLE PROCESSOR	The downloaded program is not configured for a micro controller.	If you want to use a micro controller with the program, reconfigure your controller with MPS or APS (choose Bul. 1761).
0016	STARTUP PROTECTION AFTER POWERLOSS; S:1/9 IS SET	The system has powered up in the REM Run mode. Bit S:1/13 is set and the user–fault routine is run before beginning the first scan of the program.	• Either reset bit S:1/9 if this is consistent with your application requirements, and change the mode back to REM Run, or • clear S:1/13, the major fault bit.
0018	USER PROGRAM IS INCOMPATIBLE WITH OPERATING SYSTEM	An incompatible program was downloaded. Either the program does not have the correct number of files or it does not have the correct size data files. The default program is loaded.	• Check the configuration and make sure the correct processor is selected. • If you want to use a micro controller with the program, reconfigure your controller with MPS or APS (choose Bul. 1761), or RSLogix 500.

Run Errors

Error Code (Hex)	Advisory Message	Description	Recommended Action
0004	RUNTIME MEMORY INTEGRITY ERROR	While the controller was in the RUN mode or any test mode, the ROM or RAM became corrupt. If the user program is valid, the program and initial data downloaded to the controller is used and the Retentive Data Lost Bit (S:5/8) is set. If the user program is invalid, error 0003 occurs.	• Cycle power on your unit. • Download your program and re–initialize any necessary data. • Start up your system. • Contact your local Allen–Bradley representative if the error persists.
0020	MINOR ERROR AT END OF SCAN, SEE S:5	A minor fault bit (bits 0–7) in S:5 was set at the end of scan.	Enter the status file display, clear the fault and return to REM Run mode.
0022	WATCHDOG TIMER EXPIRED, SEE S:3	The program scan time exceeded the watchdog timeout value (S:3H).	• Verify if the program is caught in a loop and correct the problem. • Increase the watchdog timeout value in the status file.
0024	INVALID STI INTERRUPT SETPOINT, SEE S:30	An invalid STI interval exists (not between 0 and 255).	Set the STI interval between the values of 0 and 255.
0025	TOO MANY JSRs IN STI SUBROUTINE	There are more than 3 subroutines nested in the STI subroutine (file 5).	Correct the user program to meet the requirements and restrictions for the JSR instruction, then reload the program and enter the REM Run mode.
0027	TOO MANY JSRs IN FAULT SUBROUTINE	There are more than 3 subroutines nested in the fault routine (file 3).	Correct the user program to meet the requirements and restrictions for the JSR instruction, then reload the program and enter the REM Run mode.
002A	INDEXED ADDRESS TOO LARGE FOR FILE	The program is referencing through indexed addressing an element beyond a file boundary.	Correct the user program to not go beyond file boundaries.
002B	TOO MANY JSRs IN HSC	There are more than 3 subroutines nested in the high–speed counter routine (file 4).	Correct the user program to meet the requirements and restrictions for the JSR instruction, then reload the program and enter the REM Run mode.
0030	SUBROUTINE NESTING EXCEEDS LIMIT OF 8	There are more than 8 subroutines nested in the main program file (file 2).	Correct the user program to meet the requirements and restrictions for the main program file, then reload the program and enter the REM Run mode.

Error Code (Hex)	Advisory Message	Description	Recommended Action
0031	UNSUPPORTED INSTRUCTION DETECTED	The program contains an instruction(s) that is not supported by the micro controller. For example MSG, SVC, or PID.	Modify the program so that all instructions are supported by the controller, then reload the program and enter the REM Run mode.
0032	SQO/SQC CROSSED DATA FILE BOUNDARIES	A sequencer instruction length/position parameter points past the end of a data file.	Correct the program to ensure that the length and position parameters do not point past the data file. Reload the program and enter the REM Run mode.
0033	BSL/BSR/FFL/FFU/LFL/LFU CROSSED DATA FILE BOUNDARIES	The length parameter of a BSL, BSR, FFL, FFU, LFL, or LFU instruction points past the end of a data file.	Correct the program to ensure that the length parameter does not point past the data file. Reload the program and enter the REM Run mode.
0034	NEGATIVE VALUE IN TIMER PRESET OR ACCUMULATOR	A negative value was loaded to a timer preset or accumulator.	If the program is moving values to the accumulated or preset word of a timer, make certain these values are not negative. Correct the program, reload, and enter the REM Run mode.
0035	ILLEGAL INSTRUCTION (TND) IN INTERRUPT FILE	The program contains a Temporary End (TND) instruction in file 3, 4, or 5 when it is being used as an interrupt subroutine.	Correct the program, reload, and enter the REM Run mode.
0037	INVALID PRESETS LOADED TO HIGH–SPEED COUNTER	Either a zero (0) or a negative high preset was loaded to counter (C5:0) when the HSC was an Up counter or the high preset was lower than or equal to the low preset when the HSC was a Bidirectional counter.	• Check to make sure the presets are valid. • Correct the program, reload, and enter the REM Run mode.
0038	SUBROUTINE RETURN INSTRUCTION (RET) IN PROGRAM FILE 2	A RET instruction is in the main program file (file 2).	Remove the RET instruction, reload the program and enter the REM Run mode.
0040	OUTPUT VERIFY WRITE FAILURE	When outputs were written and read back by the controller, the read failed. This may have been caused by noise.	• Refer to proper grounding guidelines in the *MicroLogix 1000 Programmable Controllers User Manual*, publication 1761-6.3. • Start up your system. • Contact your local Allen–Bradley representative if the error persists.

Error Code (Hex)	Advisory Message	Description	Recommended Action
0041①	EXTRA OUTPUT BIT(S) TURNED ON	An extra output bit was set when the Extra Output Select (S:0/8) bit in the status file was reset. For 16–point controllers this includes bits 6–15. For 32–point controllers this includes bits 12–15.	• Set S:0/8 or change your application to *prevent* these bits from being turned on. • Correct the program, reload, and enter the REM Run mode.

① Valid for Series A and Series C discrete controllers only.

Download Error

Error Code (Hex)	Advisory Message	Description	Recommended Action
0018	USER PROGRAM IS INCOMPATIBLE WITH OPERATING SYSTEM	An incompatible program was downloaded. Either the program does not have the correct number of files or it does not have the correct size data files. The default program is loaded.	• Check the configuration and make sure the correct processor is selected. • If you want to use a micro controller with the program, reconfigure your controller with MPS or APS (choose Bul. 1761), or RSLogix 500.

SLC Processor Faults

The processor faults are divided into the following types:

- powerup errors
- going-to-run errors
- run errors
- user program instruction errors

Powerup Errors

Error Code (Hex)	Description	Probable Cause	Recommended Action
0001	NVRAM error.	• Either noise, • lightning, • improper grounding, • lack of surge suppression on outputs with inductive loads, or • poor power source. • Loss of battery or capacitor backup.	Correct the problem, reload the program, and run. You can use the autoload feature with a memory module to automatically reload the program and enter the Run mode.
0002	Unexpected hardware watchdog timeout.	• Either noise, • lightning, • improper grounding, • lack of surge suppression on outputs with inductive loads, or • poor power source.	Correct the problem, reload the program, and run. You can use the autoload feature with a memory module to automatically reload the program and enter the Run mode.
0003	Memory module memory error. This error can also occur when going to the REM Run mode.	Memory module is corrupted.	Re-program the memory module. If the error persists, replace the memory module.
0007	Failure during memory module transfer.	Memory module is corrupted.	Re–program the memory module. If the error persists, replace the memory module.
0008	Internal software error.	An unexpected software error occurred due to: • Either noise, • lightning, • improper grounding, • lack of surge suppression on output with inductive loads, or • poor power source.	Correct the problem, reload the program, and run. You can use the autoload feature with a memory module to automatically reload the program and enter the Run mode. If the problem re–occurs, contact your RSI representative.
0009	Internal hardware error.	An unexpected hardware error occurred due to: • Either noise, • lightning • improper grounding, • lack of surge suppression on output with inductive loads, or • poor power source.	Correct the problem, reload the program, and run. You can use the autoload feature with a memory module to automatically reload the program and enter the Run mode. If the problem re–occurs, contact your A–B representative.

Going-to-Run Errors

Error Code (Hex)	Description	Probable Cause	Recommended Action
0010	The processor does not meet the required revision level.	The revision level of the processor is not compatible with the revision level for which the program was developed.	Consult your local A–B representative to purchase an upgrade kit for your processor.
0011	The executable program file number 2 is absent.	Incompatible or corrupt program is present.	Reload the program or reprogram with RSI approved APS programming software.
0012	The ladder program has a memory error.	• Either noise, • lightning, • improper grounding, • lack of surge suppression on outputs with inductive loads, or • poor power source.	Correct the problem, reload the program, and run. If the error persists, be sure to use RSI approved APS programming software to develop and load the program.
0013	• The required memory module is absent, or • S:1/10 or S:1/11 is not set as required by the program.	• Either one of the status bits is set in the program but the required memory module is absent, or • status bit S:1/10 or S:1/11 is not set in the program stored in the memory module, but it *is* set in the program in the processor memory.	• Either install a memory module in the processor, or • upload the program from the processor to the memory module.
0014	Internal file error.	• Either noise, • lightning, • improper grounding, • lack of surge suppression on outputs with inductive loads, or • poor power source.	Correct the problem, reload the program, and run. If the error persists, be sure to use RSI approved APS programming software to develop and load the program.
0015	Configuration file error.	• Either noise, • lightning, • improper grounding, • lack of surge suppression on outputs with inductive loads, or • poor power source.	Correct the problem, reload the program, and run. If the error persists, be sure to use RSI approved APS programming software to develop and load the program.
0016	Startup protection after power loss. Error condition exists at powerup when bit S:1/9 is set and powerdown occurred while running.	Status bit S:1/9 has been set by the user program. Refer to appendix A for details on the operation of status bit S:1/9.	• Either reset bit S:1/9 if this is consistent with the application requirements, and change the mode back to run, or • clear S:1/13, the major fault bit, before the end of the first program scan is reached.
0017	NVRAM/memory module user program mismatch.	Bit S:2/9 is set and the memory module user program does not match the NVRAM user program.	Transfer the memory module program to NVRAM then change to Run mode.

Error Code (Hex)	Description	Probable Cause	Recommended Action
0018	Incompatible user program. Operating system type mismatch. This error can also occur during powerup.	The user program is too advanced to be executed in the current operating system.	Contact your RSI representative for information about available operating systems for the 5/03 processor.
0019	A duplicate label number was detected.	A duplicate or missing label instruction was found in a subroutine.	• Either remove the duplicate label, or • add a label.

Run Errors

Error Code (Hex)	Description	Probable Cause	Recommended Action
001F	A program integrity problem occurred during an online editing session.	Either noise, communication loss, or a power cycle occurred during an online edit session.	Reload the program and re–enter your changes.
0004	Memory error occurred while in the Run mode.	• Either noise, • lightning, • improper grounding, • lack of surge suppression on outputs with inductive loads, or • poor power source.	Correct the problem, reload the program, and run. You can use the autoload feature with a memory module to automatically reload the program and enter the Run mode.
0020	A minor error bit is set at the end of the scan. Refer to S:5 minor error bits (lower byte only).	• Either a math or FRD instruction overflow has occurred, • sequencer or shift register instruction error was detected, • a major error was detected while executing a user fault routine, or • M0–M1 file addresses were referenced in the user program for a disabled slot.	Correct the programming problem, reload the program and enter the run mode. See also minor error bits S:5 in appendix B.

Error Code (Hex)	Description	Probable Cause	Recommended Action
0021	A remote power failure of an expansion I/O chassis has occurred. Note: A modular system that encounters an over–voltage or over–current condition in any of its power supplies can produce any of the I/O error codes listed on pages 15–16 through 15–19 (instead of code 0021). The over–voltage or over–current condition is indicated by the power supply LED being off. ⚠ Fixed and FRN 1 through 4 SLC 5/01 processors – if the remote power failure occurred while the processor was in the REM Run mode, error 0021 will cause the major error halted bit (S:1/13) to be cleared at the next powerup of the local chassis. SLC 5/02 processor and FRN 5 SLC 5/01 processors – power to the local chassis does not need to be cycled to resume the REM Run mode. Once the remote chassis is re–powered, the CPU will restart the system.	Fixed and FRN 1 to 4 SLC 5/01 processors: Power was removed or the power dipped below specification for an expansion chassis. SLC 5/02 processors and FRN 5 SLC 5/01 processors: This error code is present only while power is not applied to an expansion chassis. This is the only self-clearing error code. When power is re-applied to the expansion chassis, the fault will be cleared.	Fixed and FRN 1 to 4 SLC 5/01 processors: Cycle power on the local chassis. SLC 5/02 processors and FRN 5 SLC 5/01 processors: Re-apply power to the expansion chassis.
0022	The user watchdog scan time has been exceeded.	• Either Watchdog time is set too low for the user program, or • user program caught in a loop.	• Either increase the watchdog timeout in the status file (S:3H), or • correct the user program problem.
0023	Invalid or non-existent STI interrupt file.	• Either an STI interrupt file number was assigned in the status file, but the subroutine file was not created, or • the STI interrupt file number assigned was 0, 1, or 2.	• Either disable the STI interrupt setpoint (S:30) and file number (S:31) in the status file, or • create an STI interrupt subroutine file for the file number assigned in the status file (S:31). The file number must not be 0, 1, or 2.

Error Code (Hex)	Description	Probable Cause	Recommended Action
0024	Invalid STI interrupt interval (greater than 2550 ms or negative).	The STI setpoint is out of range (greater than 2550 ms or negative).	• Either disable the STI interrupt setpoint (S:30) and file number (S:31) in the status file, or • create an STI interrupt routine for the file number referenced in the status file (S:31). The file number must not be 0, 1, or 2.
0025	Excessive stack depth/JSR calls for the STI routine.	A JSR instruction is calling for a file number assigned to an STI routine.	Correct the user program to meet the requirements and restrictions for the JSR instruction, then reload the program and run.
0026	Excessive stack depth/JSR calls for an I/O interrupt routine.	A JSR instruction is calling for a file number assigned to an I/O interrupt routine.	Correct the user program to meet the requirements and restrictions for the JSR instruction, then reload the program and run.
0027	Excessive stack depth/JSR calls for the user fault routine.	A JSR instruction is calling for a file number assigned to the user fault routine.	Correct the user program to meet the requirements and restrictions for the JSR instruction, then reload the program and run.
0028	Invalid or non-existent “startup protection” fault routine file value.	• Either a fault routine file number was created in the status file, but the fault routine file was not physically created, or • the file number created was 0, 1, or 2.	• Either disable the fault routine file number (S:29) in the status file, or • create a fault routine for the file number referenced in the status file (S:29). The file number must not be 0, 1, or 2.
0029	Indexed address reference is outside of the entire data file space (range of B3:0 through the last file). ⚠ The SLC 5/02 processor uses an index value of zero for the faulted instruction following error recovery.	The program is referencing through indexed addressing an element beyond the allowed range. The range is from B3:0 to the last element of the last data file created by the user.	Correct and reload the user program. This problem cannot be corrected by writing to the index register word (S:24).
002A	Indexed address reference is beyond the specific referenced data file.	The program is referencing through indexed addressing an element beyond a file boundary.	Correct the user program, allocate more data space using the memory map, or re-save the program allowing crossing of file boundaries. Reload the user program. This problem cannot be corrected by writing to the index register word (S:24).
002B	An invalid file number for an indirect address exists.	The file number exists, but it is not the correct file type or the file number does not exist.	Check the file type or create the file number..

Error Code (Hex)	Description	Probable Cause	Recommended Action
002C	The referenced indirect address element is outside data file limits.	The indirectly referenced element does not exist, but the file type is correct and it exists.	Create the indirectly reference element.
002D	An invalid referenced indirect address subelement exists.	Either a subelement is referenced incorrectly or an indiret reference has been made to an M-file.	Correct the references and try again.
002E	Invalid DII Input slot.	The referenced slot is empty or a non–discrete I/O card is present.	Change the input slot to a discrete I/O card.
002F	Invalid or non-existent DII interrupt file.	• Either an DII interrupt file number was assigned in the status file, but the subroutine file was not created, or • the DII interrupt file number assigned was 0, 1, or 2.	Either disable the DII function by writing a zero to this location, or change the value to a valid ladder file (3–255).

User Program Instruction Errors

Error Code (Hex)	Description	Probable Cause	Recommended Action
0030	An attempt was made to jump to one too many nested subroutine files. This code can also mean that a program has potential recursive routines.	• Either more than the maximum of 4 (8 if you are using a 5/02 or 5/03 processor) levels of nested subroutines are called for in the user program, or • nested subroutine(s) are calling for subroutine(s) of a previous level.	Correct the user program to meet the requirements and restrictions for the JSR instruction, then reload the program and run.
0031	An unsupported instruction reference was detected.	The type or series level of the processor does not support an instruction residing in the user program, or you have programmed a constant as the first operand of a compare instruction.	• Either replace the processor with one that supports the user program, or • modify the user program so that all instructions are supported by the processor, then reload the program and run.
0032	A sequencer instruction length/position parameter points past the end of a data file.	The program is referencing an element beyond a file boundary set up by the sequencer instruction.	Correct the user program or allocate more data file space using the memory map, then reload and run.
0033	The length parameter of an LFU, LFL, FFU, FFL, BSL, or BSR instruction points past the end of a data file.	The program is referencing an element beyond a file boundary set up by the instruction.	Correct the user program or allocate more data file space using the memory map, then reload and run.

Error Code (Hex)	Description	Probable Cause	Recommended Action
0034	A negative value for a timer accumulator or preset value was detected. Fixed processors with 24 VDC input only: A negative or zero HSC preset was detected in a HSC instruction.	The accumulated or preset value of a timer in the user program was detected as being negative.	If the user program is moving values to the accumulated or preset word of a timer, make certain these values cannot be negative. Correct the user program, reload, and run.
0034 **(related to fixed 5/01 HSC instruction)**	A negative or zero HSC preset was detected in an HSC instruction.	The preset value for the HSC instruction is out of the valid range. Valid range is 1–32767.	If the user program is moving values to the preset word of the HSC instruction, make certain the values are within the valid range. Correct the user program, reload, and run.
0035	TND, SVC, or REF instruction is called within an interrupting or user fault routine.	A TND, SVC, or REF instruction is being used in an interrupt or user–fault routine. This is illegal.	Correct the user program, reload, and run.
0036	An invalid value is being used for a PID instruction parameter.	An invalid value was loaded into a PID instruction by the user program or by the user via the data monitor function for this instruction.	Code 0036 is discussed in chapter 10 in this manual.
0038	A RET instruction was detected in a non-subroutine file.	A RET instruction resides in the main program.	Correct the user program, reload, and run.
xx3A	An attempt to write to a protected data file occurred.	An attempt was made to write to an indirect address located in a file that has constant data file protection.	Remove the protection and retry the function.
1f39	An invalid string length was detected in a string file.	The first word of string data contains a negative, zero, or value greater than 82.	Check the first word of the string data elements for invalid values and correct the user data.

I/O Errors

ERROR CODES: The characters xx in the following codes represent the slot number, in hexadecimal. If the exact slot cannot be determined, the characters xx become 03 for fixed controllers and 1F for modular controllers. Refer to the table to the right.

RECOVERABLE I/O FAULTS (SLC 5/02 and higher processors only): Many I/O faults are recoverable. To recover, you must disable the specified slot, xx, in the user fault routine. If you do not disable slot xx, the processor will fault at the end of the scan.

Note: An I/O card that is severly damaged may cause the processor to indicate that an error exists in slot 1 even though the damaged card is installed in a slot other than 1.

SLOT NUMBERS (xx) IN HEXADECIMAL

Slot	xx	Slot	xx	Slot	xx	Slot	xx
0	00	8	08	16	10	24	18
1	01	9	09	17	11	25	19
2	02	10	0A	18	12	26	1A
3	03**	11	0B	19	13	27	1B
4	04	12	0C	20	14	28	1C
5	05	13	0D	21	15	29	1D
6	06	14	0E	22	16	30	1E
7	07	15	0F	23	17		1F*

Error Code (Hex)	Description	Probable Cause	Recommended Action
xx50	A chassis data error is detected.	• Either noise, • lightning, • improper grounding, • lack of surge suppression on outputs with inductive loads, or • poor power source.	Correct the problem, clear the fault, and re-enter Run mode.
xx51	A “stuck” runtime error is detected on an I/O module.	If this is a discrete I/O module, this is a noise problem. If this is a specialty I/O module, refer to the applicable user manual for the probable cause.	Cycle power to the system. If this does not correct the problem, replace the module.
xx52	A module required for the user program is detected as missing or removed.	An I/O module configured for a particular slot is missing or has been removed.	• Either disable the slot in the status file (S:11 and S:12), or • insert the required module in the slot.

Error Code (Hex)	Description	Probable Cause	Recommended Action
xx53	When going-to-run, a user program declares a slot as unused, and that slot is detected as having an I/O module inserted. This code can also mean that an I/O module has reset itself.	• Either the I/O slot is not configured for a module, but a module is present, or • the I/O module has reset itself.	• Either disable the slot in the status file (S:11 and S:12), clear the fault and run, • remove the module, clear the fault and run, or • modify the I/O configuration to include the module, then reload the program and run. • If you suspect that the module has reset itself, clear the major fault and run.
	SLC 5/03 specific – An attempt was made to enter the run or test mode with an empty chassis.	A chassis is void of all I/O modules.	Disable all slots in the empty chassis (see S:11 and S:12).
xx54	A module required for the user program is detected as being the wrong type.	An I/O module in a particular slot is a different type than was configured for that slot by the user.	• Either replace the module with the correct module, clear the fault, and run, or • change the I/O configuration for the slot, reload the program, and run.
xx55	A discrete I/O module required for the user program is detected as having the wrong I/O count. This code can also mean that a specialty card driver is incorrect.	• If this is a discrete I/O module, the I/O count is different from that selected in the I/O configuration. • If this is a specialty I/O module, the card driver is incorrect.	• If this is a discrete I/O module, replace it with a module having the I/O count selected in the I/O configuration. Then, clear the fault and run, or • change the I/O configuration to match the existing module, then reload the program and run. • If this is a specialty I/O module, refer to the user manual for that module.
xx56	The chassis configuration specified in the user program is detected as being incorrect.	The chassis configuration specified by the user does not match the hardware.	Correct the chassis configuration, reload the program and run.
xx57	A specialty I/O module has not responded to a Lock Shared Memory command within the required time limit.	The specialty I/O module is not responding to the processor in the time allowed.	Cycle chassis power. If this does not correct the problem, refer to the user manual for the specialty I/O module. You may have to replace the module.
xx58	A specialty I/O module has generated a generic fault. The card fault bit is set (1) in the module's status byte.	Refer to the user manual of the specialty I/O module.	Cycle chassis power. If this does not correct the problem, refer to the user manual for the specialty I/O module. You may have to replace the module.

Error Code (Hex)	Description	Probable Cause	Recommended Action
xx59	A specialty I/O module has not responded to a command as being completed within the required time limit.	A specialty I/O module did not complete a command from the processor in the time allowed.	Refer to the user manual for the specialty I/O module. You may have to replace the module.
xx5A	Hardware interrupt problem.	If this is a discrete I/O module, this is a noise problem. If this is a specialty I/O module, refer to the user manual for the module.	Cycle chassis power. Check for a noise problem and be sure proper grounding practices are used. If this is a specialty I/O module, refer to the user manual for the module. You may have to replace the module.
xx5B	G file configuration error – user program G file size exceeds the capacity of the module.	G file is incorrect for the module in this slot.	Refer to the user manual for the specialty I/O module. Reconfigure the G file as directed in the manual, then reload and run.
xx5C	M0–M1 file configuration error – user program M0–M1 file size exceeds capacity of the module.	M0–M1 files are incorrect for the module in this slot.	Refer to the user manual for the specialty I/O module. Reconfigure the M0–M1 files as directed in the manual, then reload and run.
xx5D	Interrupt service requested is not supported by the processor.	The specialty I/O module has requested service and the processor does not support it.	Refer to the user manual for the specialty I/O module to determine which processors support use of the module. Change processor to one that supports the module.
xx5E	Processor I/O driver (software) error.	Corrupt processor I/O driver software.	Reload program using RSI approved APS software.
xx60 through xx6F	Identifies an I/O module specific recoverable major error.	–	–
xx70 through xx7F	Identifies an I/O module specific non-recoverable major error.	–	–
xx80 through xx8F	Identifies a specialty I/O module specific non-recoverable major error.	–	–
xx90	Interrupt problem on a disabled slot.	A specialty I/O module requested service while a slot was disabled.	Refer to the user manual for the specialty I/O module. You may have to replace the module.
xx91	A disabled slot has faulted.	A specialty I/O module in a disabled slot has faulted.	Cycle chassis power. If this does not correct the problem, refer to the user manual for the specialty I/O module. You may have to replace the module.

Error Code (Hex)	Description	Probable Cause	Recommended Action
xx92	Invalid or non-existent module interrupt subroutine (ISR) file.	The I/O configuration/ISR file information for a specialty I/O module is incorrect.	Correct the I/O configuration/ISR file information for the specialty I/O module. Refer to the user manual for the module for the correct ISR file information. Then reload the program and run.
xx93	Unsupported I/O module specific major error.	The processor does not recognize the error code from a specialty I/O module.	Refer to the user manual for the specialty I/O module.
xx94	A module has been detected as being inserted under power in the run or test mode. This can also mean that an I/O module has reset itself.	The module was inserted in the chassis under power, or the module has reset itself.	No module should ever be inserted in a chassis under power. If this occurs and the module is not damaged, • Either remove the module, clear the fault and run, or • add the module to the I/O configuration, reference the module in the user program where required, reload the program, and run.
0x01A0	A major fault unique to the SLC 5/05. The error code indicates communication channel hardware fault has occured.	Ethernet communication fault.	The fault may be cleared via a write to the System Status File, but Ethernet communications will be disabled until a power cycle is performed. Word 15 of the System Status File provides a specific fault code for the Ethernet daughterboard when user fault code 0x01A0 is generated. Recoverable, but Ethernet communication is disabled.

Troubleshooting SLC 5/03 and Higher Processors

Between the time you apply power to the processor, and it has a chance to establish communication with a connected programming device, the only form of communication between you and the processor is through the LED display.

Powerup LED Display

When power is applied, all the LEDs flash on momentarily and then off. This is part of the normal power-up sequence. Following the self test by the processor, all of the LEDs flash on again momentarily. If a user program is in a running state, the RUN LED is illuminated. If a fault exists within the processor, the FLT LED is illuminated.

Identifying Processor Errors while Downloading an Operating System

The download process takes approximately 45 seconds. During this time, watch the LED display for status information. While the download is in progress, the RUN and FLT LEDs remain off. The RS232, DH485 or DH+, Ethernet, FORCE, and BATT LEDs illuminate in a pre-defined sequence. If the download is successful, the above LEDs are illuminated.

If during the download process of an operating system type memory module or during the normal power-up self test process an error occurs, the FLT LED is illuminated and the four LEDs flash on and off at a rate of 2 seconds.

The following table describes the possible LED combinations that are displayed every other time the LEDs flash on.

ON LED Display	Description
FAULT, FORCE, DH485, DH+, or Ethernet	Fatal hardware error exists.
FAULT, FORCE, RS232, DH485 or DH+	A hardware watchdog timeout exists.
FAULT, BATT	NVRAM error exists.
FAULT, BATT, RS232	The contents of the operating system memory module are corrupt.
FAULT, BATT, DH485 or DH+	The downloadable operating system is not compatible with the hardware.
FAULT, BATT, RS232, DH485, DH+, or Ethernet	An attempt was made to download the operating system onto write–protected memory.
FAULT, BATT, FORCE	Flash EEPROM failure.
FAULT, BATT, FORCE, RS232	Failure during transmission of downloadable operating system.
FAULT, BATT, FORCE, DH485, DH+, or Ethernet	The operating system is missing or has been corrupted.

A MicroLogix 1000 Controller Status File

This appendix discusses the status file functions of MicroLogix 1000 controllers.

Status File Overview

The status file lets you monitor how your operating system works and lets you direct how you want it to work. This is done by using the status file to set up control bits and monitor both hardware and software faults and other status information.

Note *Do not write to reserved words in the status file. If you intend writing to status file data, it is imperative that you first understand the function fully.*

The status file S: contains the following words:

Word	Function	Page
S:0	Arithmetic Flags	A–3
S:1L (low byte)	Controller Mode Status/Control (low)	A–5
S:1H (high byte)	Controller Mode Status/Control (Hi)	A–5
S:2L (low byte)	Controller Alternate Mode Status/Control (low)	A–8
S:2H (high byte)	Controller Alternate Mode Status/Control (Hi)	A–8
S:3L (low byte)	Current Scan Time	A–11
S:3H (high byte)	Watchdog Scan Time	A–11
S:4	Timebase	A–12
S:5	Minor Error Bits	A–12
S:6	Major Error Code	A–14
S:7	Suspend Code	A–18
S:8 to S:12	Reserved	A–18
S:13, S:14	Math Register	A–18
S:15L (low byte)	DF1 Node Address	A–18
S:15H (high byte)	DF1 Baud Rate	A–19
S:16L (low byte)	DH-485 Node Address	A–19
S:16H (high byte)	DH-485 Baud Rate	A–19
S:17 to S:21	Reserved	A–19
S:22	Maximum Observed Scantime	A–19
S:23	Reserved	A–20
S:24	Index Register	A–20
S:25 to S:29	Reserved	A–20
S:30	STI Setpoint	A–20
S:31 and S:32	Reserved	A–20

Status File Descriptions

The following tables describe the status file functions, beginning at address S:0 and ending at address S:32.

Each status bit is classified as one of the following:

- **Status** — Use these words, bytes, or bits to monitor controller operation or controller status information. The information is seldom written to by the user program or programming device (unless you want to reset or clear a function such as a monitor bit).
- **Dynamic Configuration** — Use these words, bytes, or bits to select controller options while online with the controller.
- **Static Configuration** — Use these words, bytes, or bits to select controller options while in the offline program mode, prior to downloading the user program.

Address	Bit	Classification	Description
S:0	Arithmetic and Scan Status Flags		The arithmetic flags are assessed by the controller following the execution of certain math and data handling instructions. The state of these bits remain in effect until certain math or data handling instructions in the program are executed.
S:0/0	Carry	Status	This bit is set by the controller if a mathematical carry or borrow is generated. Otherwise the bit remains cleared. This bit is assessed as if a function of unsigned math. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/0 is restored when execution resumes.
S:0/1	Overflow	Status	This bit is set by the controller when the result of a mathematical operation does not fit in its destination. Otherwise the bit remains cleared. Whenever this bit is set, the overflow trap bit S:5/0 is also set except for the ENC bit. Refer to S:5/0. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/1 is restored when execution resumes.

Address	Bit	Classification	Description
S:0/2	Zero	Status	This bit is set by the controller when the result of certain math or data handling instructions is zero. Otherwise the bit remains cleared. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/2 is restored when execution resumes.
S:0/3	Sign	Status	This bit is set by the controller when the result of certain math or data handling instructions is negative. Otherwise the bit remains cleared. When a STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of S:0/3 is restored when execution resumes.
S:0/4 to S:0/7	Reserved		
S:0/8①	Extend I/O Configuration	Static Configuration	This bit must be set by the user when unused outputs are written to. If reset and unused outputs are turned on the controller will fault (41H).
S:0/9	Reserved	NA	NA
S:0/10	Primary Protocol	Static Configuration	This bit defines the protocol that the controller will initially use when attempting to establish communication, where: 0 = DF1 (default setting) 1 = DH-485
S:0/11	Active Protocol	Status	This bit is updated by the controller during a protocol switch. It indicates which protocol is currently being used for communication, where: 0 = DF1 1 = DH-485
S:0/12	Selected DF1 Protocol	Status	This bit allows the user to determine which DF1 protocol is configured, where: 0 = DF1 Full-Duplex (default setting) 1 = DF1 Half-Duplex
S:0/13 to S:0/15	Reserved	NA	NA

① Valid for Series A–C discrete only.

Address	Bit	Classification	Description
S:1/0 to S:1/4	Controller Mode Status/ Control	Status	Bits 0–4 function as follows: 0 0000 = (0) Remote Download in progress 0 0001 = (1) Remote Program mode 0 0011 = (3) Suspend Idle (operation halted by SUS instruction execution) 0 0110 = (6) Remote Run mode 0 0111 = (7) Remote Test continuous mode 0 1000 = (8) Remote Test single scan mode
S:1/5	Forces Enabled	Status	This bit is set by the controller (1) to indicate that forces are always enabled.
S:1/6	Forces Installed	Status	This bit is set by the controller to indicate that forces have been set by the user.
S:1/7	Comms Active	Status	This bit is set when the controller receives valid data from the programming port. For DF1 protocols, the bit is reset if the controller does not receive valid data from the programming port for 10 seconds. **Note:** In DF1 half-duplex mode, simple polls by the DF1 master or replies to received messages will not reset the timer. A poll with a command is required to reset the timer. For DH-485, the bit is reset as soon as the DH-485 link layer determines that no other devices are active on the link. **Application Note:** For DF1 half-duplex, you can use this bit to enable a timer (via an XIO instruction) to sense whether the DF1 master is actively communicating to the slave. The preset of the timer is determined by the total network timing.
S:1/8	Fault Override at Powerup	Static Configuration	When set, this bit causes the controller to clear the Major Error Halted bit S:1/13 and Minor error bits S:5/0 to S:5/7 on power up if the processor had previously been in the REM Run mode and had faulted. The controller then attempts to enter the REM Run mode. Set this bit using the Data Monitor function offline only.

Address	Bit	Classification	Description
S:1/9	Startup Protection Fault	Static Configuration	When this bit is set and power is cycled while the controller is in the REM Run mode, the controller executes the user–fault routine prior to the execution of the first scan of your program. You have the option of clearing the Major Error Halted bit S:1/13 to resume operation in the REM Run mode. If user–fault routine does not reset bit S:1/13, the fault mode results. Program the user-fault routine logic accordingly. When executing the startup protection fault routine, S:6 (major error fault code) will contain the value 0016H.
S:1/10 to S:1/11	Reserved		
S:1/12	Run Always	Static Configuration	When set, this bit causes the controller to clear S:1/13 and S:5/0–7 before attempting to enter RUN mode when power is applied or if an unexpected reset occurs. If this bit is not set, the controller powers up in the previous mode it was in before losing power, unless the controller was in REM test mode. If the controller was in REM test mode when power was removed, the controller enters REM program mode when power is applied. This bit overrides any faults existing at power down. ⚠ **Setting the Run Always bit causes the controller to enter the REM Run mode if an unexpected reset occurs, regardless of the mode that the controller was in before the reset occurred. Unexpected resets may occur due to electromagnetic noise, improper grounding, or an internal controller hardware failure. Make sure your application is designed to safely handle this situation.**

Address	Bit	Classification	Description
S:1/13	Major Error Halted	Dynamic Configuration	This bit is set by the controller any time a major error is encountered. The controller enters a fault condition. Word S:6, the Fault Code will contain a code that can be used to diagnose the fault condition. Any time bit S:1/13 is set, the controller: • either places all outputs in a safe state (outputs are off) and energizes the fault LED, • or enters the user–fault routine with outputs active (if in REM Run mode), allowing the fault routine ladder logic to attempt recovery from the fault condition. If the user–fault routine determines that recovery is required, clear S:1/13 using ladder logic prior to exiting the fault routine. If the fault routine ladder logic does not understand the fault code, or if the routine determines that it is not desirable to continue operation, the controller exits the fault routine with bit S:1/13 set. The outputs are placed in a safe state and theFAULT LED is energized. When you clear bit S:1/13 using a programming device, the controller mode changes from fault to Remote Program. You can move a value to S:6, then set S:1/13 in your ladder program to generate an application-specific major error. All application-generated faults are recoverable regardless of the value used. **Note:** *Once a major fault state exists, you must correct the condition causing the fault, and you must also clear this bit in order for the controller to accept a mode change attempt (into REM Run or REM Test). Also, clear S:6 to avoid the confusion of having an error code but no fault condition.* **Note:** *Do not re-use error codes that are defined later in this appendix as application-specific error codes. Instead, create your own unique codes. This prevents you from confusing application errors with system errors. We recommend using error codes FFOO to FFOF to indicate application-specific major errors.*

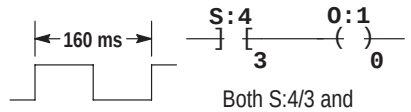
Address	Bit	Classification	Description
S:1/14	OEM Lock	Static Configuration	Using this bit you can control access to a controller file. To program this feature, select "Future Access Disallow" when saving your program. When this bit is cleared, it indicates that any compatible programming device can access the ladder program (provided that password conditions are satisfied).
S:1/15	First Pass	Status	Use this bit to initialize your program as the application requires. When this bit is set by the controller, it indicates that the first scan of the user program is in progress (following power up in the RUN mode or entry into a REM Run or REM Test mode). The controller clears this bit following the first scan. This bit is set during execution of the startup protection fault routine. Refer to S:1/9 for more information.
S:2/0	STI Pending	Status	When set, this bit indicates that the STI timer has timed out and the STI routine is waiting to be executed. This bit is cleared upon starting the STI routine, ladder program, exit of the REM Run or Test mode, or execution of a true STS instruction.
S:2/1	STI Enabled	Status and Static Configuration	This bit may be set or reset using the STS, STE, or STD instruction. If set, it allows execution of the STI if the STI setpoint S:30 is non–zero. If clear, when an interrupt occurs, the STI subroutine does not execute and the STI Pending bit is set. The STI Timer continues to run when this bit is disabled. The STD instruction clears this bit. If this bit is set or reset editing the status file online, the STI is not affected. If this bit is set, the bit allows execution of the STI. If this bit is reset editing the status file offline, the bit disallows execution of the STI.
S:2/2	STI Executing	Status	When set, this bit indicates that the STI timer has timed out and the STI subroutine is currently being executed. This bit is cleared upon completion of the STI routine, ladder program, or REM Run or Test mode.
S:2/3 to S:2/4	Reserved	NA	NA

Address	Bit	Classification	Description
S:2/5[1]	Incoming Command Pending Bit	Status	This bit is set when the processor determines that another node on the network has requested information or supplied a command to it. This bit can be set at any time. This bit is cleared when the processor services the request (or command).
S:2/6[1]	Message Reply Pending Bit	Status	This bit is set when another node on the network has supplied the information you requested in the MSG instruction of your processor. This bit is cleared when the processor stores the information and updates your MSG instruction.
S:2/7[1]	Outgoing Message Command Pending Bit	Status	This bit is set when one or more messages in your program are enabled and waiting, but no message is being transmitted at the time. As soon as transmission of a message begins, the bit is cleared. After transmission, the bit is set again if there are further messages waiting. It remains cleared if there are no further messages waiting.
S:2/8 to S:2/13	Reserved	NA	NA

[1] Valid for Series C discrete only.

Address	Bit	Classification	Description
S:2/14	Math Overflow Selection	Dynamic Configuration	Set this bit when you intend to use 32-bit addition and subtraction. When S:2/14 is set, and the result of an ADD, SUB, MUL, or DIV instruction cannot be represented in the destination address (underflow or overflow), • the overflow bit S:0/1 is set, • the overflow trap bit S:5/0 is set, • and the destination address contains the unsigned truncated least significant 16 bits of the result. The default condition of S:2/14 is reset (0). When S:2/14 is reset, and the result of an ADD, SUB, MUL, or DIV instruction cannot be represented in the destination address (underflow or overflow), • the overflow bit S:0/1 is set, • the overflow trap bit S:5/0 is set, • and the destination address contains 32767 if the result is positive or – 32768 if the result is negative. Note, the status of bit S:2/14 has no effect on the DDV instruction. Also, it has no effect on the math register content when using MUL and DIV instructions. To program this feature, use the Data Monitor function to set or clear this bit. To provide protection from inadvertent data monitor alteration of your selection, program an unconditional OTL instruction at address S:2/14 to ensure the new math overflow operation. Program an unconditional OTU instruction at address S:2/14 to ensure the original math overflow operation.
S:2/15	Reserved		

Address	Bit	Classification	Description
S:3L	Current Scan Time	Status	The value of this byte tells you how much time elapses in a program cycle. A program cycle includes: • scanning the ladder program, • housekeeping, • scanning the I/O, • servicing of the communication channel. The byte value is zeroed by the controller each scan, immediately preceding the execution of rung 0 of program file 2 (main program file). The byte is incremented every 10 ms thereafter, and indicates, in 10 ms increments, the amount of time elapsed in each scan. If this value ever equals the value in S:3H Watchdog, a user watchdog major error will be declared (code 0022). The resolution of the scan time value is +0 to 90 ms (–10 ms). Example: The value 9 indicates that 80–90 ms has elapsed since the start of the program cycle.
S:3H	Watchdog Scan Time	Dynamic Configuration	This byte value contains the number of 10 ms ticks allowed to occur during a program cycle. The default value is 10 (100 ms), but you can increase this to 255 (2.55 seconds) or decrease it to 1, as your application requires. If the program scan S:3L value equals the watchdog value, a watchdog major error will be declared (code 0022).

Address	Bit	Classification	Description
S:4	Timebase	Status	All 16 bits of this word are assessed by the controller. The value of this word is zeroed upon power up in the REM Run mode or entry into the REM Run or REM Test mode. It is incremented every 10 ms thereafter. Application note: You can write any value to S:4. It will begin incrementing from this value. You can use any individual bit of this word in your user program as a 50% duty cycle clock bit. Clock rates for S:4/0 to S:4/15 are: 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240, 20480, 40960, 81920, 163840, 327680, and 655360 ms. The application using the bit must be evaluated at a rate more than two times faster than the clock rate of the bit. In the example below, bit S:4/3 toggles every 80 ms, producing a 160 ms clock rate. To maintain accuracy of this bit in your application, the instruction using bit S:4/3 (O:1/0 in this case) must be evaluated at least once every 79.999 ms.  S:4/3 cycles in 160 ms Both S:4/3 and Output O:1/0 toggle every 80 ms. O:1/0 must be evaluated at least once every 79.999 ms.
S:5	Minor Error Bits		The bits of this word are set by the controller to indicate that a minor error has occurred in your ladder program. Minor errors, bits 0 to 7, revert to major error 0020H if any bit is detected as being set at the end of the scan. These bits are automatically cleared on a power cycle.
S:5/0	Overflow Trap	Dynamic Configuration	When this bit is set by the controller, it indicates that a mathematical overflow has occurred in the ladder program. See S:0/1 for more information. If this bit is ever set upon execution of the END or TND instruction, major error (0020) is declared. To avoid this type of major error from occurring, examine the state of this bit following a math instruction (ADD, SUB, MUL, DIV, DDV, NEG, SCL, TOD, or FRD), take appropriate action, and then clear bit S:5/0 using an OTU instruction with S:5/0.

Address	Bit	Classification	Description
S:5/1	Reserved		
S:5/2	Control Register Error	Dynamic Configuration	The LFU, LFL, FFU, FFL, BSL, BSR, SQO, SQC, and SQL instructions are capable of generating this error. When bit S:5/2 is set, it indicates that the error bit of a control word used by the instruction has been set. If this bit is ever set upon execution of the END or TND instruction, major error (0020) is declared. To avoid this type of major error from occurring, examine the state of this bit following a control register instruction, take appropriate action, and then clear bit S:5/2 using an OTU instruction with S:5/2.
S:5/3	Major Error Detected While Executing user–fault routine	Dynamic Configuration	When set, the major error code (S:6) represents the major error that occurred while processing the fault routine due to another major error.
S:5/4 to S:5/7	Reserved		
S:5/8	Retentive Data Lost	Status	This bit is set whenever retentive data is lost. This bit remains set until you clear it. While set, this bit causes the controller to fault prior to the first true scan of the program.
S:5/9	Reserved		
S:5/10	STI Lost	Status	This bit is set whenever the STI timer expires while the STI routine is either executing or disabled *and* the pending bit (s:2/0) is already set.
S:5/11 to S:5/12	Reserved		
S:5/13	Input Filter Selection Modified	Status	This bit is set whenever the discrete input filter selection in the controller is made compatible with the hardware.

Address	Bit	Classification	Description
S:5/14 to S:5/15	Reserved		
S:6	Major Error Code	Status	A hexadecimal code is entered in this word by the controller when a major error is declared. Refer to S:1/13. The code defines the type of fault, as indicated on the following pages. This word is not cleared by the controller. Error codes are presented, stored, and displayed in a hexadecimal format. If you enter a fault code as a parameter in an instruction in your ladder program, you must convert the code to decimal. **Application note**: You can declare your own application-specific major fault by writing a unique value to S:6 and then setting bit S:1/13. Interrogate the value of S:6 in the user–fault routine to determine the type of fault that occurred. Fault Classifications: Faults are classified as Non-User, Non-Recoverable, and Recoverable. Error code descriptions and classifications are listed on the following pages. Categories are: • powerup errors • going–to–run errors • run errors • download errors

Each fault is classified as one of the following:

- **Non–User** — A fault caused by various conditions that cease ladder program execution. The user-fault routine is not run when this fault occurs.
- **Non–Recoverable** — A fault caused by the user that cannot be recovered from. The user-fault routine is run when this fault occurs. However, the fault cannot be cleared.
- **Recoverable** — A fault caused by the user that can be recovered from in the user–fault routine by resetting major error halted bit (S:1/13). The user-fault routine is run when this fault occurs.

Address	Error Code (Hex)	Powerup Errors	Fault Classification		
			Non-User	User	
				Non-Recoverable	Recoverable
S:6	0001	The default program was loaded.	X		
	0002	Unexpected reset occurred.	X		
	0003	EEPROM memory is corrupt.	X		
	0008	A fatal internal software error occurred.	X		
	0009	A fatal internal hardware error occurred.	X		

Address	Error Code (Hex)	Going–to–Run (GTR) Errors①	Fault Classification		
			Non-User	User	
				Non-Recoverable	Recoverable
S:6	0005	Retentive data is lost.			X
	0010	The downloaded program is not a controller program.	X		
	0016	Startup protection after power loss, S:1/9 is set. The user must check for a retentive data lost condition if the user–fault routine was executed with startup protection.			X

① Going–to–Run errors occur when the controller is going from any mode to REM Run mode or from any non–Run mode (PRG, SUS) to Test mode.

Address	Error Code (Hex)	Run Errors	Fault Classification		
			Non-User	User	
				Non-Recoverable	Recoverable
S:6	0004	A runtime memory integrity error occurred.	X		
	0020	A minor error at the end of the scan. Refer to S:5.			X
	0022	The watchdog timer expired. Refer to S:3H.		X	
	0024	Invalid STI interrupt setpoint. Refer to S:30.		X	
	0025	There are excessive JSRs in the STI subroutine (file 5).		X	
	0027	There are excessive JSRs in the fault subroutine (file 3).		X	
	002A	The indexed address is too large for the file.		X	
	002B	There are excessive JSRs in the high-speed counter sub-routine (file 4).		X	
	0030	The subroutine nesting exceeds a limit of 8 (file 2).		X	
	0031	An unsupported instruction was detected.	X		
	0032	An SQO/SQC instruction crossed data file boundaries.			X
	0033	The LFU, LFL, FFU, FFL, BSL, or BSR instruction crossed data file boundaries.			X
	0034	A negative value for a timer accumulator or preset value was detected.			X
	0035	An illegal instruction (TND) occurred in the interrupt file.		X	
	0037	Invalid presets were loaded to the high-speed counter.			X
	0038	A RET instruction was detected in program file 2.	X		

Address	Error Code (Hex)	Run Errors	Fault Classification		
			Non-User	User	
				Non-Recoverable	Recoverable
S:6	0040	An output verify write occurred.		X	
	0041[1]	Extra output bit(s) turned on.		X	

[1] Valid for Series A–C discrete only.

Address	Error Code (Hex)	Download Errors	Fault Classification		
			Non-User	User	
				Non-Recoverable	Recoverable
S:6	0018	The user program is incompatible with the operating system.	X		

Address	Bit	Classification	Description
S:7	Suspend Code	Status	When a non-zero value appears in S:7, it indicates that the SUS instruction identified by this value has been evaluated as true, and the Suspend Idle mode is in effect. This pinpoints the conditions in the application that caused the Suspend Idle mode. This value is not cleared by the controller. Use the SUS instruction with startup troubleshooting, or as runtime diagnostics for detection of system errors.
S:8 to S:12	Reserved		
S:13 and S:14	Math Register	Status	Use this double register to produce 32–bit signed divide and multiply operations, precision divide or double divide operations, and 5–digit BCD conversions. These two words are used in conjunction with the MUL, DIV, DDV, FRD, and TOD math instructions. The math register value is assessed upon execution of the instruction and remains valid until the next MUL, DIV, DDV, FRD, or TOD instruction is executed in the user program. An explanation of how the math register operates is included with the instruction definitions. If you store 32-bit signed data values, you must manage this data type without the aid of an assigned 32-bit data type. For example, combine B3:0 and B3:1 to create a 32-bit signed data value. We recommend that you start all 32-bit values on an even or odd word boundary for ease of application and viewing. Also, we recommend that you design, document, and view the contents of 32-bit signed data in either the hexadecimal or binary radix. When an STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of the math register is restored when execution resumes.
S:15L	DF1 Node Address	Status	This byte value contains the node address of your processor on the DF1 link. It is used when executing Message (MSG) instructions over the DF1 link. The default node address of a processor is 1. Valid node addresses are 0–254. To change a processor node address you must use a programming device.

Address	Bit	Classification	Description
S:15H	DF1 Baud Rate	Status	This byte value contains a code used to select the baud rate of the processor on the DF1 link. The controller baud rate options are: • 300 • 600 • 1200 • 2400 • 4800 • 9600 (default) • 19200 • 38400 To change the baud rate from the default value you must use a programming device.
S:16L	DH-485 Node Address	Status	This byte value contains the node address of your processor on the DH-485 link. Each device on the DH-485 link must have a unique address between the decimal values 1–31. To change a processor node address, you must use a programming device.
S:16H	DH-485 Baud Rate	Status	This byte value contains the baud rate of the processor on the DH-485 link. The controller baud rate options are: • 9600 • 19200 (default) To change the baud rate from the default value, you must use a programming device.
S:17 to S:21	Reserved	NA	NA
S:22	Maximum Observed Scantime	Dynamic Configuration	This word indicates the maximum observed interval between consecutive program cycles. This value indicates, in 10 ms increments, the time elapsed in the longest program cycle of the controller. Refer to S:3L for more information regarding the program cycle. The controller compares each last scan value to the value contained in S:22. If the controller determines that the last scan value is larger than the value stored at S:22, the last scan value is written to S:22. Resolution of the maximum observed scan time value is +0 to –10 ms. For example, the value 9 indicates that 80–90 ms was observed as the longest program cycle. Interrogate this value using the Data Monitor function if you need to determine or verify the longest scan time of your program.

Address	Bit	Classification	Description
S:23	Reserved		
S:24	Index Register	Status	This word indicates the element offset used in indexed addressing. When an STI, high-speed counter, or Fault Routine interrupts normal execution of your program, the original value of this register is restored when execution resumes.
S:25 to S:29	Reserved		
S:30	STI Setpoint	Dynamic Configuration	You enter the timebase to be used in the selectable timed interrupt (STI). The time can range from 10 to 2550 ms. (This is in 10ms increments, so valid values are from 0–255.) Your STI routine executes per the value you enter. Write a zero value to disable the STI. To provide protection from inadvertent Data Monitor alteration of your selection, program an unconditional MOV instruction containing the setpoint value of your STI to S:30, or program a CLR instruction at S:30 to prevent STI operation. If the STI is initiated while in the REM Run mode by loading the status registers, the interrupt starts timing from the end of the program scan in which the status registers were loaded.
S:31 to S:32	Reserved		

B SLC Status File

This appendix lists the:

- SLC processor status file overview
- status file detailed word/bit descriptions

This appendix discusses the status file functions of the Fixed, SLC 5/01, SLC 5/02, SLC 5/03, SLC 5/04 and SLC 5/05 processors. The processors function similarly, but the higher numbered processors utilize more features. The tables in this appendix indicate which functions are supported by each processor.

The appendix starts with an overview listing of the status file. A more detailed description of each status word follows. Use the overview list to find the page number of the detailed description.

Status File Overview

The status file lets you monitor how your operating system works and lets you direct how you want it to work. This is done by using the status file to set up interrupts, load memory module programs, and monitor both hardware and software faults.

Note *Do not write to status file data unless the word or bit is listed as dynamic/static configuration in the descriptions that follow. If you intend writing to status file data, it is imperative that you first understand the function fully.*

The status file S: contains the following words:

Word	Function	Applies To	Page
S:0	Arithmetic and Scan Status Flags	all processors	B–5
S:1L	Processor Mode Status/Control		B–6
S:1H	Processor Mode Status/Control		B–6
S:2	Processor Alternate Mode Status/Control		B–12
S:3L	Current Scan Time		B–18
S:3H	Watchdog Scan Time		B–19
S:4	Free Running Clock		B–20
S:5	Minor Error Bits		B–21
S:6	Major Error Fault Code		B–24
S:7, S:8	Suspend Code/Suspend File		B–36
S:9	Channel 1 Active Nodes (SLC 5/03) Unused (SLC 5/04) Ethernet Daughterboard Firmware Series (SLC 5/05)		B–37
S:10	Channel 1 Active Nodes (SLC 5/03) Unused (SLC 5/04) Ethernet Daughterboard Firmware Revision (SLC 5/05)		B–37
S:11, S;12	I/O Slot Enables		B–37
S:13, S:14	Math Register		B–38
S:15L	Channel 1 DH-485 Node Address (SLC 5/03) Channel 1 DH+ Node Address (SLC 5/04) Ethernet Daughterboard Fault Code (SLC 5/05)		B–39
S:15H	Channel 1 DH-485 Baud Rate (SLC 5/03) Channel 1 DH+ Baud Rate (SLC 5/04) Ethernet Daughterboard Fault Code (SLC 5/05)		B–40
S:16, S:17	Word Single Step Rung/File	SLC 5/02 and higher	B–41
S:18, S:19	Single Step Breakpoint Rung/File		B–42
S:20, S:21	Word Fault Powerdown Rung/File		B–43
S:22	Maximum Observed Scan Time		B–44
S:23	Average Scan Time		B–44

Word	Function	Applies To	Page
S:24	Index Register	SLC 5/02 and higher	B–45
S:25, S:26	I/O Interrupt Pending		B–45
S:27, S28	I/O Interrupt Enabled		B–46
S:29	User Fault Routine File Number		B–47
S:30	Selectable Timed Interrupt Set Point		B–47
S:31	Selectable Timed Interrupt File Number		B–48
S:32	I/O Interrupt Executing		B–48
S:33	Extended Processor Status and Control		B–48
S:34	Processor Extended Mode Status/Control (SLC 5/04 and higher)		B–54
S:35	Last 1 ms Scan Time		B–56
S:36	Extended Minor Error Bits Reserved		B–56
S:37	Clock/Calendar Year		B–57
S:38	Clock/Calendar Month		B–57
S:39	Clock/Calendar Day		B–57
S:40	Clock/Calendar Hours		B–58
S:41	Clock/Calendar Minutes		B–58
S:42	Clock/Calendar Seconds		B–58
S:43	Selectable Timed Interrupt Time (SLC 5/03 and higher)		B–58
S:44	I/O Event Interrupt Time (SLC 5/03 and higher)		B–58
S:45	Discrete Input Interrupt Time (SLC 5/03 and higher)		B–58
S:46	Discrete Input Interrupt File Number		B–58
S:47	Discrete Input Interrupt Input Slot		B–59
S:48	Discrete Input Interrupt Bit Mask		B–59
S:49	Discrete Input Interrupt Compare Value	SLC 5/03 and higher	B–60
S:50	Discrete Input Interrupt Down Count		B–60
S:51	Discrete Input Interrupt Return Mask		B–61
S:52	Discrete Input Interrupt Accumulator		B–61
S:53L	Day-of-Week (SLC 5/03 OS302 Series B and higher, SLC 5/04 OS401 Series B and higher, and SLC 5/05)		B–61
S:53H and S:54	Reserved		B–61
S:55	Last DII ISR Scan Time		B–61
S:56	Maximum DII ISR Scan Time		B–62
S:57	Operating System Catalog Number		B–62
S:58	Operating System Series		B–62
S:59	Operating System FRN		B–62

Word	Function	Applies To	Page
S:60	Processor Catalog Number		B–62
S:61	Processor Series		B–62
S:62	Processor Revision	SLC 5/03 and higher	B–62
S:63	User Program Type		B–62
S:64	User Program Functionality Index		B–63
S:65	User RAM Size		B–63
S:66	Flash EEPROM Size		B–63
S:67 to S:82	Channel 0 Active Node Table		B–63
S:83 to S:86	Channel 1 Active Node Table	SLC 5/04	B–63
S:87 to S:98	Reserved		B–63
S:99	Global Status Word		B–64
S:100 to S:163	Global Status File		B–64

Status File Details

Conventions Used in the Displays

The following tables describe the status file functions, beginning at address S:0 and ending at address S:163. A bullet (•) indicates that the function applies to the specified processor.

The following classifications are used:

- **Status** – Use these words, bytes, or bits to monitor processor options or processor status information. The information is seldom written to the user program or programming device (unless you want to reset or clear a function such as a minor error bit).
- **Dynamic Configuration** – Use these words, bytes, or bits to select processor options while in the RUN mode.
- **Static Configuration** – Use these words, bytes, or bits to select processor options prior to entering the RUN mode. Note that some options must be selected while in the offline program mode, prior to restoring the user program.

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:0		**Arithmetic and Scan Status Flags** The arithmetic flags are assessed by the processor following the execution of any math, logical, or move instruction. The state of these bits remains in effect until the next math, logical, or move instruction in the program is executed.	•	•	•	•	•
S:0/0	Status	**Carry Bit** This bit is set by the processor if a mathematical carry or borrow is generated. Otherwise the bit remains cleared. This bit is assessed as if a function of unsigned math.	•	•	•	•	•
		When a STI, I/O Slot, or Fault Routine interrupts normal execution of your program, the original value of S:0/0 is restored when execution resumes.		•	•	•	•
		When a DII interrupts normal execution of your program, the original value of S:0/0 is restored when execution resumes.			•	•	•
S:0/1	Status	**Overflow Bit** This bit is set by the processor when the result of a mathematical operation does not fit in its destination. Otherwise the bit remains cleared. Whenever this bit is set, the overflow trap bit S:5/0 is also set. Refer to S:5/0.	•	•	•	•	•
		When a STI, I/O Slot, or Fault Routine interrupts normal execution of your program, the original value of S:0/1 is restored when execution resumes.		•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:0/1 continued		When a DII interrupts normal execution of your program, the original value of S:0/1 is restored when execution resumes.			•	•	•
S:0/2	**Status**	**Zero Bit** This bit is set by the processor when the result of a math, logical, or move instruction is zero. Otherwise the bit remains cleared.	•	•	•	•	•
		When a STI, I/O Slot, or Fault Routine interrupts normal execution of your program, the original value of S:0/2 is restored when execution resumes.		•	•	•	•
		When a DII interrupts normal execution of your program, the original value of S:0/2 is restored when execution resumes.			•	•	•
S:0/3	**Status**	**Sign Bit** This bit is set by the processor when the result of a math, logical, or move instruction is negative. Otherwise the bit remains cleared.	•	•	•	•	•
		When a STI, I/O Slot, or Fault Routine interrupts normal execution of your program, the original value of S:0/3 is restored when execution resumes.		•	•	•	•
		When a DII interrupts normal execution of your program, the original value of S:0/3 is restored when execution resumes.			•	•	•
S:0/4 to S:0/15	**NA**	**Reserved**	•	•	•	•	•
S:1/0 to S:1/4	**Status**	**Processor Mode Status/Control** Bits 0-4 function as follows: 0 0000 = (0) Remote Download in progress 0 0001 = (1) Remote Program mode (the fault mode exists when bit S:1/13 is set along with mode 0 0001) 0 0011 = (3) Suspend Idle (operation halted by SUS instruction execution) fault mode exists when bit S:1/13 is set along with mode 0 0011 0 0110 = (6) Remote Run mode 0 0111 = (7) Remote Test continuous mode 0 1000 = (8) Remote Test single scan mode 0 1001 = (9) Remote Test single step (step until) **Note:** *All modes in the fixed, SLC 5/01, and SLC 5/02 processors are considered as remote because they do not have a keyswitch.*	•	•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:1/0 to S:1/4 continued		1 0000 = (16) Download in progress (keyswitch=PROGram) 1 0001 = (17) PROGram mode – the fault mode exists when bit S:1/13 is set along with mode 1 0001 1 1011 = (27) Suspend Idle – the fault mode exists when bit S:1/13 is set along with mode 1 1011 (keyswitch=RUN) 1 1110 = (30) RUN – the fault mode exists when bit S:1/13 is set along with mode 1 1110 (keyswitch = RUN) All other values for bits 0-4 are reserved.			•	•	•
S:1/5	Status	**Forces Enabled Bit** This bit is set by the processor if you have enabled forces in a ladder program. Otherwise, the bit remains cleared. The processor Forced I/O LED is on continuously when forces are enabled.	•	•	•	•	•
S:1/6	Status	**Forces Installed Bit** This bit is set by the processor if you have installed forces in a ladder program. The forces may or may not be enabled. Otherwise the bit remains cleared. The processor Forced I/O LED flashes when forces are installed, but not enabled.	•	•	•	•	•
S:1/7	Status	**Communications Active Bit (Channel 1)** This bit is set by the processor when at least one other node is present on the network attached to channel 1. Otherwise, the bit remains cleared. When the node is active, it is a recognized participant in a DH-485 or DH+ token-passing network. For Ethernet communications, this bit is only an indication that the Ethernet daughter board is functioning properly, not necessarily that there are any other active Ethernet nodes, or that channel 1 is connected to an Ethernet network.	•	•	•	•	•
S:1/8	Dynamic Config	**Fault Override at Powerup Bit** When set, this bit causes the processor to clear the Major Error Halted bit S:1/13 and Minor error bits S:5/0 to S:5/7 on power up; if the processor had previously been in the REM Run mode and had faulted. The processor then attempts to enter the REM Run mode. When this bit remains cleared (default value), the processor remains in a major fault state at power up. To program this feature, set this bit using the Data Monitor function.	•	•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:1/9	Dynamic Config	**Startup Protection Fault Bit** When this bit is set and power is cycled while the processor is in the REM Run mode, the processor executes your fault routine prior to the execution of the first scan of your program. You then have the option of clearing the Major Error Halted bit S:1/13 to resume operation in the REM Run mode. If your fault routine does not reset bit S:1/13, the fault mode results. To program this feature, use the Data Monitor function, then program your fault routine logic accordingly. When executing the startup protection fault routine, S:6 (major error fault code) will contain the value 0016H.		•	•	•	•
S:1/10	Static Config	**Load Memory Module on Memory Error Bit** You can use this bit to transfer a memory module program to the processor in the event that a processor memory error is detected at power-up. A memory error means the processor cannot run the program in the RAM because the program has been corrupted, as detected by a parity or checksum error. This type of error is caused by battery or capacitor drain, noise, or a power problem. You *must* set S:1/10 in the status file of the program in the memory module. When a memory module is installed that has bit S:1/10 set, a processor memory error detected at power-up causes the memory module program to be transferred to the processor, and the REM Run mode to be entered. When S:1/10 is cleared in the memory module, the processor remains in a major fault condition if a memory error is detected on power-up, regardless if a memory module exists. When S:1/10 is set in the status file of the user program in RAM memory, the memory module *must* be installed at all times to enter the REM Run or REM Test modes. To program this feature, set this bit using the Data Monitor function. Then store the program in the memory module.	•	•	•	•	•

<table>
<tr><th>Address</th><th>Classification</th><th>Description</th><th>Fixed 5/01</th><th>5/02</th><th>5/03</th><th>5/04</th><th>5/05</th></tr>
<tr><td>S:1/11</td><td>Static Config</td><td>Load Memory Module Always Bit
When this bit is set, you can overwrite a processor program with a memory module program by cycling processor power. A programming device is not required. The processor mode after powerup is as follows for SLC 5/02 and higher processors:
<table><tr><th>Mode before Powerdown</th><th>Mode after Powerup</th></tr><tr><td>REM Test/Program</td><td>REM Program</td></tr><tr><td>REM Run</td><td>REM Run</td></tr><tr><td>Fault after REM Test/Program</td><td>REM Program REM Run</td></tr><tr><td>Fault after REM Run</td><td>REM Program</td></tr><tr><td>REM Idle</td><td>REM Program</td></tr><tr><td>REM Download</td><td></td></tr></table></td><td></td><td>•</td><td>•</td><td>•</td><td>•</td></tr>
<tr><td></td><td></td><td><table><tr><th>Mode before Powerdown</th><th>Mode after Powerup (same keyswitch position)</th></tr><tr><td>Run</td><td>RUN</td></tr><tr><td>Program</td><td>PROGram</td></tr><tr><td>Idle</td><td>RUN</td></tr><tr><td>Fault after Run</td><td>RUN</td></tr><tr><td>Fault after Program</td><td>PROGram</td></tr></table></td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr>
<tr><td></td><td></td><td>Note: All modes in the fixed, SLC 5/01, and SLC 5/02 processors are considered to be remote because they do not have a keyswitch.
The memory module you install in the processor must have status file bit S:1/11 set. Loading takes place if the master password and/or password in the processor and memory module match. Loading can also take place if the processor has neither a password nor master password.
When S:1/11 is also set in the status file of the user program in RAM, the memory module must be installed at all times to enter the REM Run or REM Test modes.
⚠ The overwriting process, including data tables, is repeated each time you cycle power.
To program this feature, set this bit using the Data Monitor function. Then store the program in the memory module.</td><td></td><td>•</td><td>•</td><td>•</td><td>•</td></tr>
<tr><td></td><td></td><td>You may choose not to overwrite data files on a per file basis.</td><td></td><td></td><td>•</td><td>•</td><td>•</td></tr>
</table>

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:1/12	Static Config	**Load Memory Module and Run Bit** With this bit, you can overwrite a processor program with a memory module program by cycling processor power. A programming device is not required. The processor will attempt to enter the REM Run mode, regardless of what mode was in effect before cycling power:		•	•	•	•
		Mode before Powerdown / **Mode after Powerup** REM Test/Rem Program → REM Run REM Run/Rem Fault → REM Run REM Idle/Rem Download → REM Run		•	•	•	•
		Mode before Powerdown / **Mode after Powerup (same keyswitch position)** Run → RUN Idle → Run Program/Download → PROGram Fault after Run → RUN Fault after Program → PROGram			•	•	•
		Note: *All modes in the fixed, SLC 5/01, and SLC 5/02 processors are considered to be remote because they do not have a keyswitch.* The memory module you install in the processor must have status file bit S:1/12 set. Loading takes place if the master password and/or password in the processor and memory module match. Loading can also take place if the processor has neither a password nor master password. When S:1/12 is set in the status file of the user program in RAM, it does not require the presence of the memory module to enter the REM Run or REM Test mode. **Application example:** Set both S:1/11 and S:1/12 to autoload and run every power cycle, and require the presence of the memory module to enter the REM Run or REM Test modes. ⚠ **If you leave the memory module installed, the overwriting process, including data tables, is repeated each time you cycle power. The mode is changed to REM Run each and every power cycle.** To program this feature, use the Data Monitor function. Then store the program in the memory module. This feature is particularly useful when you are troubleshooting hardware failures with "spares" (replacement modules). Use this feature to facilitate application logic upgrades in the field without a programming device.		•	•	•	•
		You may choose not to overwrite data files on a per file basis.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:1/13	Dynamic Config	**Major Error Halted Bit** This bit is set by the processor any time a major error is encountered. The processor enters a fault condition. Word S:6, Fault Code will contain a code which can be used to diagnose the fault condition. Any time bit S:1/13 is set, the processor: • either places all outputs in a safe state and energizes the fault LED, or	•	•	•	•	•
		• enters the user fault routine with outputs active, allowing the fault routine ladder logic to attempt recovery from the fault condition. If your fault routine determines that recovery is required, clear S:1/13 using ladder logic prior to exiting the fault routine. If the fault routine ladder logic does not understand the fault code, or if the routine determines that it is not desirable to continue operation, exit the fault routine with bit S:1/13 set. The outputs will be placed in a safe state and the fault LED will be energized.		•	•	•	•
		When you clear bit S:1/13 using a programming device, the processor mode changes from fault to either Remote Program, or Remote Idle Suspend depending on the previous mode of the processor. You can move a value to S:6, then set S:1/13 in your ladder program to generate an application specific Major Error. **Note**: *Once a major fault state exists, you must correct the condition causing the fault, and you must also clear this bit in order for the processor to accept a mode change attempt (into REM Program, REM Run, or REM Test). Also, clear S:6 to avoid the confusion of having an error code but no fault condition.* **Note:** *Do not re-use error codes that are defined in the SLC error code list in chapter 15 as application specific error codes. Instead, create your own unique codes. This prevents you from confusing application errors with system errors. We recommend using error codes FFOO to FFOF to indicate application specific major errors.*	•	•	•	•	•
		When you clear bit S:1/13 using a programming device, the processor mode changes from fault to either Program, Run, or Idle Suspend depending on the previous mode of the processor. You can move a value to S:6, then set S:1/13 in your ladder program to generate an application specific major error. ⚠ **If you clear this bit with the keyswitch in RUN, the processor immediately enters the RUN mode.** You can clear faults S:1/13 and S:6 by cycling the keyswitch to PROGram and then to RUN.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:1/14	Status	**Access Denied Bit (OEM Lock)** You can allow or deny future access to a processor file. Set this bit to deny access. This indicates that a programming device must have a matching copy of the processor file in its memory in order to monitor the ladder program. A programming device that does not have a matching copy of the processor file is denied access. To program this feature, select "Future Access Disallow" when saving your program. To provide protection from inadvertent data monitor alteration of your selection, program an unconditional OTL instruction at address S:1/14, to deny future access. Program an unconditional OTU instruction at address S:1/14 to allow future access. When this bit is cleared, it indicates that any compatible programming device can access the ladder program (provided that password conditions are satisfied). When access is denied, the programming device (APS or HHT) may not access the ladder program. Functions such as change mode, clear memory, restore program, and transfer memory module are allowed regardless of this selection. A device such as the DTAM is not affected by this function.	•	•	•	•	•
S:1/15	Status	**First Pass Bit** Use this bit to initialize your program as the application requires. When this bit is set by the processor, it indicates that the first scan of the user program is in progress (following power up in the RUN mode or entry into a REM Run or REM Test mode). The processor clears this bit following the first scan. When this bit is cleared, it indicates that the program is not in the first scan of a REM Test or REM Run mode.	•	•	•	•	•
		This bit is set during execution of the startup protection fault routine. Refer to S:1/9 for more information.		•	•	•	•
S:2/0	Status	**STI (Selectable Timed Interrupt) Pending Bit** When set, this bit indicates that the STI timer has timed out and the STI routine is waiting to be executed. This bit is cleared upon starting of the STI routine, power up, exit of the REM Run mode, or execution of a true STS instruction.		•	•	•	•
		The STI pending bit will not be set if the STI timer expires while executing the fault routine.		•			
		This bit is set if the STI timer expires while executing the DII subroutine or fault routine.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:2/1	Static Config	**STI (Selectable Timed Interrupt) Enabled Bit** This bit is set in its default condition, or when set by the STE or STS instruction. If set, it allows execution of the STI if the STI file (S:31) and STI setpoint (S:30) are non-zero. If clear, when an interrupt occurs, the STI subroutine does not execute and the STI Pending bit is set. The STI Timer continues to run when disabled. The STD instruction clears this bit.		•	•	•	•
	Dynamic Config	Use the Data Monitor function to set and clear this bit, or address this bit with your ladder logic program. This bit is set in its default condition, or when set by the STE or STS instruction. If set, it allows execution of the STI if the STI file (word 31) and STI rate (word 30) are non-zero. If clear, the STI subroutine does not execute and the STI pending bit is set. The STI timer continues to run. The STD instruction clears this bit.			•	•	•
S:2/2	Status	**STI (Selectable Timed Interrupt) Executing Bit** When set, this bit indicates that the STI timer has timed out and the STI subroutine is currently being executed. This bit is cleared upon completion of the STI routine, powerup, or REM Run mode entry. **Application example:** You can examine this bit in your fault routine to determine if your STI was executing when the fault occurred.		•	•	•	•
S:2/3	Static Config	**Index Addressing File Range Bit** When clear, the index register can only index within the same data file of the specified base address. When set, the index register can index anywhere from data file B3:0 to the end of the last declared data file. This bit is selected at the time you save your program.		•	•	•	•
		The SLC 5/03 and higher processors allow you to index from 0:0 to the last data file. **Note:** *Change this bit while in the offline mode only. Save the program after changing the bit.*			•	•	•
S:2/4	Static Config	**Saved with Single Step Test Enabled Bit** When clear, the Single Step Test mode function is not available. Clear also indicates that debug registers S:16 through S:21 are inoperative. When set, the program can operate in the Single Step Test mode. See descriptions of S:16 through S:21. When set, your program requires 0.375 instruction words (3 bytes) per rung of additional memory. This bit is selected at the time you save your program.		•			
		Note: *This bit is not applicable to the SLC 5/03 and higher processors since its functionality is always available and requires no special compile time selection.*			•	•	•

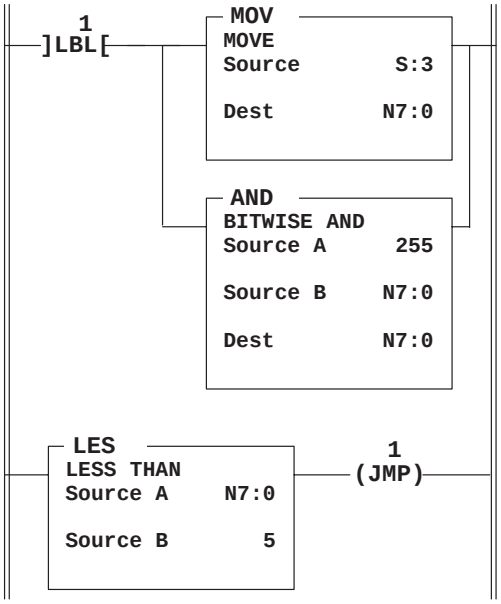
Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:2/5	Status	**Incoming Command Pending Bit (Channel 1)** This bit is set when the processor determines that another node on the network has requested information or supplied a command to it. This bit can be set at any time. This bit is cleared when the processor services the request (or command). Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.		•	•	•	•
S:2/6	Status	**Message Reply Pending Bit (Channel 1)** This bit is set when another node on the network has supplied the information you requested in the MSG instruction of your processor. This bit is cleared when the processor stores the information and updates your MSG instruction. Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.		•	•	•	•
S:2/7	Status	**Outgoing Message Command Pending Bit (Channel 1)** This bit is set when one or more messages in your program are enabled and waiting, but no message is being transmitted at the time. As soon as transmission of a message begins, the bit is cleared. After transmission, the bit is set again if there are further messages waiting. It remains cleared if there are no further messages waiting. Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.		•	•	•	•
S:2/8	Dynamic Config	**CIF (Common Interface File) Addressing Mode** This bit controls the mode used to address elements in the CIF file (data file 9) when processing a communication request. Word address mode – in effect when the bit is clear (0): This is the default setting, compatible with other SLC 500 devices on the DH-485 network. Byte address mode – in effect when the bit is set (1): This mode is used when the processor is receiving a message from a device on the network, possibly through a bridge or gateway. This setting is compatible with Allen-Bradley PLC inter-processor communication.		•	•	•	•

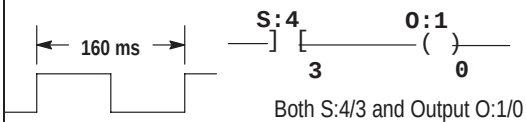
Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:2/9	Static Config	**Memory Module Program Compare** When this bit is set inside a valid program that is contained in a memory module, no modification of the NVRAM user program files is allowed. This includes online editing, program downloading, and clear memory commands. Use this feature to prevent a programming device from altering the NVRAM program from the program contained in the Memory Module. If a memory module is installed with this bit set, and a different NVRAM user program is contained in NVRAM, the processor will not enter the Run mode. You must transfer the memory module program to NVRAM in order to enter the Run mode.			•	•	•
S:2/10	Static Config	**STI Resolution Selection (1 ms or 10 ms) Bit** This bit is cleared by default. When clear, this bit uses a 10 ms timebase for the STI Setpoint (S:30) value. For example, the value 4 uses a 40 ms STI setpoint. When set, this bit uses a 1 ms timebase for the STI Setpoint (S:30). For example, the value 4 uses a 4 ms STI setpoint. To program this feature, use the Data Monitor function to set, clear, or address this bit with your ladder program.			•	•	•
S:2/11	Status	**Discrete Input Interrupt Pending Bit** When set, this bit indicates that the DII accumulator (S:52) equals the DII preset (S:50) and the ladder file number specified by the DII file number (S:46) is waiting to be executed. It is cleared when the DII file number (S:46) begins executing, or on exit of the REM Run or REM Test mode.			•	•	•
S:2/12	Dynamic Config	**Discrete Input Interrupt Enabled Bit** To program this feature, use the Data Monitor function to set, clear, or address this bit with your ladder program. This bit is set in its default condition. If set, it allows execution of the DII Subroutine if the DII file (S:46) is non-zero. If clear, when the interrupt occurs, the DII subroutine does not execute and the DII Pending bit is set. The DII function continues to run anytime the DII file (S:46) is non-zero. If the pending bit is set, the enable bit is examined at the next end of scan.			•	•	•
S:2/13	Status	**Discrete Input Interrupt Executing Bit** When set, this bit indicates that the DII interrupt has occurred and the DII subroutine is currently being executed. This bit is cleared on completion of the DII routine, power up, or REM Run mode entry. **Application example:** You can examine this bit in your fault routine to determine if your DII was executing when the fault occurred.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:2/14	Dynamic Config	**Math Overflow Selection Bit** Set this bit when you intend to use 32-bit addition and subtraction. When S:2/14 is set, and the result of an ADD, SUB, MUL, or DIV instruction cannot be represented in the destination address (underflow or overflow), • the overflow bit S:0/1 is set, • the overflow trap bit S:5/0 is set, and • the destination address contains the unsigned truncated least significant 16 bits of the result. The default condition of S:2/14 is reset (0). When S:2/14 is reset, and the result of an ADD, SUB, MUL, or DIV instruction cannot be represented in the destination address (underflow or overflow), • the overflow bit S:0/1 is set, • the overflow trap bit S:5/0 is set, and • the destination address contains 32767 if the result is positive or – 32768 if the result is negative. **Note:** *The status of bit S:2/14 has no effect on the DDV instruction. Also, it has no effect on the math register content when using MUL and DIV instructions.* To program this feature, use the Data Monitor function to set or clear this bit. To provide protection from inadvertent data monitor alteration of your selection, program an unconditional OTL instruction at address S:2/14 to ensure the new math overflow operation. Program an unconditional OTU instruction at address S:2/14 to ensure the original math overflow operation. See chapter 3 in this manual for an application example of 32-bit signed math.		•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:2/15	Dynamic Config	**Communications Servicing Selection Bit (Ethernet Channel 1 for SLC 5/05) (DH+ Channel 1 for SLC 5/04) (DH485 Channel 1 for SLC 5/03)** When set, only one communication request/command can be serviced per END, TND, REF, or SVC. When clear, all serviceable incoming or outgoing communication requests/commands can be serviced per END, TND, REF, or SVC. When clear, communication throughput increases. However, your scan time will increase if several communication requests/commands are received in the same scan. One communication request/command consists of either an incoming command, a message reply, or an outgoing message command. See S:2/5, S:2/6, and S:2/7 and S:33/7. To program this feature, use the Data Monitor function to set or clear this bit. To provide protection from inadvertent data monitor alteration of your selection, program an unconditional OTL instruction at address S:2/15 to ensure one request/command operation, or program an unconditional OTU instruction at address S:2/15 to ensure multiple request/command operation. Alternately, your program may change the state of this bit using ladder logic if your application requires dynamic selection of this function. **Application example:** Suppose you have a system consisting of an SLC 5/03 processor, an APS programmer, and a DTAM. The program scan time for your user program is extremely long. Because of this, the programming device or DTAM takes an unusually long time to update its screen. Improve this update time by clearing S:2/15. In a case such as this, the additional time spent by the processor to service all communication at the end of the scan is insignificant compared to the time it takes to complete one scan. You could increase communication throughput even further by using an SVC instruction. See chapter 8 in this manual for more information.		•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:3L	Status	**Current/Last 10 ms Scan Time** The value of this byte tells you how much time elapses in a program cycle. A program cycle includes: • scanning the ladder program, • housekeeping, • scanning the I/O, and • servicing of the communication port. The byte value is zeroed by the processor each scan, immediately preceding the execution of rung 0 of program file 2 (main program file) or on return from the REF instruction. The byte is incremented every 10 ms thereafter, and indicates, in 10 ms increments, the amount of time elapsed in each program cycle. If this value ever equals the value in S:3H Watchdog, a user watchdog major error will be declared (code 0022). The resolution of the scan time value is +0 to –10 ms. Example: The value 9 indicates that 80–90 ms has elapsed since the start of the program cycle.	•	•			
		Note: *When SVC or REF instructions are contained in your program, this value will appear to be erratic when you monitor it with a programming device. This is because the SVC or REF instructions allow this value to be read in mid-scan, while it is still incrementing.*		•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:3L continued		**Application example:** Your application requires that each and every program scan execute in the same length of time. You measure the maximum and minimum scan times and find them to be 40 ms and 20 ms. You can make every scan equal to precisely 50 ms by programming the following rungs as the last rungs of your program.  This example assumes that your I/O scan and communication servicing takes less than 10 ms. If it exceeds 10 ms, the resolution of +0 to –1 tick (10 ms) must be added to the scan time.					
S:3H	Dynamic Config	**Watchdog Scan Time Byte** This byte value contains the number of 10 ms ticks allowed to occur during a program cycle. The default value is 10 (100 ms), but you can increase this to 250 (2.5 seconds) or decrease it to 2, as your application requires. If the program scan S:3L value equals the watchdog value, a watchdog major error will be declared (code 0022). This value is applied each END, TND, or REF.	•	•	•	•	•

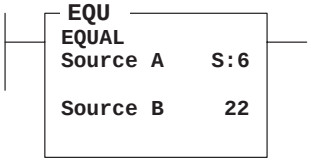
Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:4	Status	**Free Running Clock** Only the first 8 bits (byte value) of this word are assessed by the processor. This value is zeroed at powerup in the REM Run mode. You can use any individual bit of this byte in your user program as a 50% duty cycle clock bit. Clock rates for S:4/0 to S:4/7 are: 20, 40, 80, 160, 320, 640, 1280, and 2560 ms. The application using the bit must be evaluated at a rate more than two times faster than the clock rate of the bit. This is illustrated in the following example for SLC 5/02 and higher processors.	•				
	Dynamic Config	All 16 bits of this word are assessed by the processor. The value of this word is zeroed upon power up in the REM Run mode or entry into the REM Test mode. It is incremented every 10 ms thereafter. **Application note**: You can write any value to S:4. It will begin incrementing from this value. You can use any individual bit of this word in your user program as a 50% duty cycle clock bit. Clock rates for S:4/0 to S:4/15 are: 20, 40, 80, 160, 320, 640, 1280, 2560, 5120, 10240, 20480, 40960, 81920, 163840, 327680, and 655360 ms The application using the bit must be evaluated at a rate more than two times faster than the clock rate of the bit. In the following example, bit S:4/3 toggles every 80 ms, producing a 160 ms clock rate. To maintain accuracy of this bit in your application, the instruction using bit S:4/3 (O:1/0 in this case) must be evaluated at least once every 79.999 ms.  S:4/3 cycles in 160 ms Both S:4/3 and Output O:1/0 toggle every 80 ms. O:1/0 must be evaluated at least once every 79.999 ms.		•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:5		**Minor Error Bits** The bits of this word are set by the processor to indicate that a minor error has occurred in your ladder program. Minor errors, bits 0 to 7, revert to major error 0020H if any bit is detected as being set at the end of the scan. HHT users: If the processor faults for error code 0020H, you must clear minor error bits S:5/0–7 along with S:1/13 to attempt error recovery.	•	•	•	•	•
S:5/0	Dynamic Config	**Overflow Trap Bit** When this bit is set by the processor, it indicates that a mathematical overflow has occurred in the ladder program. See S:0/1 for more information. If this bit is ever set upon execution of the END, TND, or REF instruction, major error (0020) will be declared. To avoid this type of major error from occurring, examine the state of this bit following a math instruction (ADD, SUB, MUL, DIV, DDV, NEG, SCL, TOD, or FRD), take appropriate action, and then clear bit S:5/0 using an OTU instruction with S:5/0 or a CLR instruction with S:5.	•	•	•	•	•
S:5/1	NA	**Reserved**	•	•	•	•	•
S:5/2	Dynamic Config	**Control Register Error Bit** The LFU, LFL, FFU, FFL, BSL, BSR, SQO, SQC, and SQL instructions are capable of generating this error. When bit S:5/2 is set, it indicates that the error bit of the control instruction has been set. If this bit is ever set upon execution of the END, TND, or REF instruction, major error (0020) will be declared. To avoid this type of major error from occurring, examine the state of this bit following a control register instruction, take appropriate action, and then clear bit S:5/2 using an OTU instruction with S:5/2 or a CLR instruction with S:5.	•	•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:5/3	Dynamic Config	**Major Error Detected while Executing User Fault Routine Bit** When set, the major error code (S:6) represents the major error that occurred while processing the fault routine due to another major error. If this bit is ever set upon execution of the END, TND, or REF instruction, major error (0020) will be declared. To avoid this type of major error from occurring, examine the state of this bit inside your fault routine, take appropriate action, and then clear bit S:5/3 using an OTU instruction with S:5/3 or a CLR instruction with S:5. **Application example:** Suppose you are executing your fault routine for fault code 0016H Startup Protection. At rung 3 inside this fault routine, a TON containing a negative preset is executed. When rung 4 is executed, fault code 0016H is overwritten to indicate code 0034H, and S:5/3 is set. If your fault routine did not determine that S:5/3 was set, major error 0020H would be declared at the end of the first scan. To avoid this problem, examine S:5/3, followed by S:6, prior to returning from your fault routine. If S:5/3 is set, take appropriate action to remedy the fault, then clear S:5/3.		•	•	•	•
S:5/4	Dynamic Config	**M0–M1 Referenced on Disabled Slot Bit** This bit is set whenever any instruction references an M0 or M1 module file element for a slot that is disabled (via its I/O slot enable bit). When set, the bit indicates that an instruction could not execute properly due to the unavailability of the addressed M0 or M1 data. If this bit is ever set upon execution of the END, TND, or REF instruction, major error (0020) is declared. To avoid this type of major error from occurring, examine the state of this bit following a M0–M1 referenced instruction, take appropriate action, and then clear bit S:5/4 using an OTU instruction with S:5/4 or a CLR instruction with S:5.		•	•	•	•
S:5/5 to S:5/7	NA	**Reserved** Reserved for minor errors that revert to major errors at the end of the scan.	•	•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:5/8	Status	**Memory Module Boot Bit** When this bit is set by the processor, it indicates that a memory module program has been transferred to the processor. This bit is not cleared by the processor. Your program can examine the state of this bit on entry into the REM Run mode to determine if the memory module content has been transferred. Bit S:1/15 will be set to indicate REM Run mode entry. This information is useful when you have an application that contains retentive data and a memory module that has only bit S:1/10 set (Load Memory Module on Memory error). Use this bit to indicate that retentive data has been lost. This bit is also helpful when using bits S:1/11 (Load Memory Module Always) or S:1/12 (Load Memory Module Always and Run) to distinguish a power up REM Run mode entry from a REM Program (or REM Test) mode to REM Run mode entry.	•	•	•	•	•
S:5/9	Status	**Memory Module Password Mismatch Bit** This bit is set on REM Run mode entry, whenever loading from the memory module is specified (word 1, bits 11 or 12) and the processor user program is password protected, and the memory module program does not match that password. Use this bit to inform your application program that an autoloading memory module is installed but did not load due to a password mismatch.	•	•	•	•	•
S:5/10	Status	**STI (Selectable Timed Interrupt) Overflow Bit** This bit is set whenever the STI timer expires while the STI routine is either executing or disabled *and* the pending bit is already set.		•	•	•	•
S:5/11	Status	**Battery Low Bit** This bit is set whenever the Battery Low LED is on. The bit is cleared when the Battery Low LED is off.		•	•	•	•
S:5/12	Status	**Discrete Input Interrupt Overflow Bit** This bit is set whenever the DII interrupt occurs while still executing the DII subroutine or whenever the DII interrupt occurs while pending or disabled.			•	•	•
S:5/13	Dynamic Config	**Unsuccessful Operating System Load Was Attempted** This bit is set whenever an operating system memory module load is attempted and is unsuccessful. Unsuccessful loads can occur when either the protection jumper is in the protect position or is missing, or if the operating system memory module is incompatible with the SLC 5/03, SLC 5/04, or SLC 5/05 processors' hardware platform. Examine the state of this bit with your user program to diagnose this condition.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:5/14	Status	**Channel 0 Modem Lost** This bit indicates the status of the modem connected to Channel 0 (RS232 serial port). The state of the bit is determined by: • the protocol Channel 0 is configured for • the Control Line selected • the states of DCD (Data Carrier Detect) and DSR (Data Set Ready) If the bit is set, then the modem is not properly connected to Channel 0 or it is in a state where unreliable communication exchanges may take place via Channel 0. The following conditions apply: • If Channel 0 is disabled or configured for DH485, the bit is always cleared. • If Channel 0 is configured for one of the DF1 protocols in System Mode or Generic ASCII in User Mode, then the Control Line selection determines how DCD and DSR affect the modem status: – If Control Line = NO HANDSHAKING: The bit is always set. – If Control Line = FULL-DUPLEX or HALF-DUPLEX WITHOUT CONTINUOUS CARRIER: The bit is set if DSR goes inactive and cleared when DSR goes active. (DCD has no affect on modem status in this case.) – If Control Line = HALF-DUPLEX WITH CONTINUOUS CARRIER: The bit is set if either DSR goes inactive or DCD remains inactive for more than 10 seconds. This bit is cleared when both DSR and DCD go active.			•	•	•
S:5/15	Status	**ASCII String Manipulation Error** This bit applies to SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors. This bit is set to 1 when an attempt is made to process a string using an ASCII instruction that exceeds 82 characters in length.			•	•	•
S:6	Status	**Major Error Fault Code** A hexadecimal code is entered in this word by the processor when a major error is declared. Refer to S:1/13. The code defines the type of fault, as indicated on the following pages. This word is not cleared by the processor. Error codes are presented, stored, and displayed in a hexadecimal format. Refer to appendix G for more information on the hexadecimal numbering system.	•	•	•	•	•

<table>
<tr><th>Address</th><th>Classification</th><th>Description</th><th>Fixed 5/01</th><th>5/02</th><th>5/03</th><th>5/04</th><th>5/05</th></tr>
<tr><td>S:6 continued</td><td></td><td>If you enter a fault code as a parameter in an instruction in your ladder program, you must convert the code to decimal. For example, if you program an EQU instruction to go true when the error 0016 occurs, enter S:6 as source A and 22, the decimal equivalent of 0016H, as source B:
 
</td><td></td><td>•</td><td>•</td><td>•</td><td>•</td></tr>
<tr><td></td><td></td><td>Application example: You can declare your own application specific major fault by writing a unique value to S:6 and then setting bit S:1/13.
SLC 5/02 processor users: Interrogate the value of S:6 in your fault routine to determine the type of fault that occurred. If your program was saved with the test single step enabled, you can also interrogate S:20 and S:21 to pinpoint the exact rung that was executing when the fault occurred.
Fault Classifications: Faults are classified as Non-User, Non-Recoverable, and Recoverable.
<table>
<tr><th>Non-User Fault</th><th>Non-Recoverable User Fault</th><th>Recoverable User Fault</th></tr>
<tr><td>The fault routine does not execute.</td><td>The fault routine executes for 1 pass. (You may initiate a MSG instruction to another node to identify the fault condition of the processor.)</td><td>The fault routine may clear the fault by clearing bit S:1/13.</td></tr>
</table></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>Error code descriptions and classifications are listed on pages B–26 through B–35. Categories are:
• powerup errors
• going-to-run errors
• runtime errors
• user program instruction errors
• I/O errors
See chapter 15 of this manual for fault cause and recovery information.</td><td>•</td><td>•</td><td>•</td><td>•</td><td>•</td></tr>
</table>

Address	Error Code (Hex)	Powerup Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	0001	NVRAM error.	X			●	●	●	●	●
	0002	Unexpected hardware watchdog timeout.	X			●	●	●	●	●
	0003	Memory module memory error. This error can also occur while going into the REM Run mode.	X				●	●	●	●
	0005	Reserved			X			●	●	●
	0006	Reserved			X			●	●	●
	0007	Failure during memory module transfer.	X					●	●	●
	0008	Internal software error.	X					●	●	●
	0009	Internal hardware error.	X					●	●	●

Address	Error Code (Hex)	Going-to-Run Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	0010	The Processor does not meet the required revision level.	X			•	•	•	•	•
	0011	The executable program file number 2 is absent.	X			•	•	•	•	•
	0012	The ladder program has a memory error.	X			•	•	•	•	•
	0013	• The required memory module is absent or • S:1/10 or S:1/11 is not set as required by the program.			X	•	•	•	•	•
	0014	Internal file error.	X			•	•	•	•	•
	0015	Configuration file error.	X			•	•	•	•	•
	0016	Startup protection after power loss. Error condition exists at powerup when bit S:1/9 is set and powerdown occurred while running.			X		•	•	•	•
	0017	NVRAM/memory module user program mismatch.		X				•	•	•
	0018	Incompatible user program – Operating system type mismatch. This error can also occur during powerup.	X					•	•	•
	0019	Missing or duplicate label was detected.		X				•	•	•
	001F	A program integrity problem occurred during an online editing session.	X					•	•	•

Address	Error Code (Hex)	Runtime Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	0004	Memory error occurred while in the Run mode.	X				•	•	•	•
	0020	A minor error bit is set at the end of the scan. Refer to S:5 minor error bits.			X	•	•	•	•	•
	0021	Remote power failure of an expansion I/O chassis occurred. **Note**: *A modular system that encounters an over–voltage or over–current condition in any of its power supplies can produce any of the I/O error codes listed on pages B–33 and B–35 (instead of code 0021). The over–voltage or over–current condition is indicated by the power supply LED being off.* ⚠ **Fixed and FRN 1 to 4 SLC 5/01 processors – If the remote power failure occurred while the processor was in the REM Run mode, error 0021 will cause the major error halted bit (S:1/13) to be cleared at the next powerup of the local chassis.** **SLC 5/02 processors and FRN 5 SLC 5/01 processors – Power to the local chassis does not need to be cycled to resume the REM Run mode. Once the remote chassis is re–powered, the CPU will restart the system.**	X			•	•	•	•	•
	0022	The user watchdog scan time has been exceeded.		X		•	•	•	•	•
	0023	Invalid or non-existent STI interrupt file.		X			•	•	•	•

Address	Error Code (Hex)	Runtime Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	0024	Invalid STI interrupt interval (greater than 2559 ms or negative).		X			•	•	•	•
	0025	Excessive stack depth/JSR calls for STI routine.		X			•	•	•	•
	0026	Excessive stack depth/JSR calls for I/O interrupt routine.		X			•	•	•	•
	0027	Excessive stack depth/JSR calls for user fault routine.		X			•	•	•	•
	0028	Invalid or non-existent “startup protection” fault routine file value.	X				•	•	•	•
	0029	Indexed address reference outside of entire data file space (range of B3:0 through the last file).			X		•			
		⚠ **The SLC 5/02 processor uses an index value of zero for the faulted instruction following error recovery.**		X				•	•	•
	002A	Indexed address reference is beyond specific referenced data file.		X			•	•	•	•
	002B	The file number exists, but it is not the correct file type or the file number does not exist.			X			•	•	•
	002C	The indirectly referenced element does not exist, but the file type is correct and it exists. For example, T4:[N7:0] N7:0=10, but T4 only goes to T4:9.			X			•	•	•
	002D	Either a subelement is referenced incorrectly or an indirect reference has been made to an M-file.			X			•	•	•

Address	Error Code (Hex)	Runtime Errors	Fault Classification			Processor				
			Non-User	User		Fixed 5/01	5/02	5/03	5/04	5/05
				Non-Recov	Recov					
S:6 continued	002E	Invalid DII Input slot.			X			•	•	•
	002F	Invalid or non-existent DII interrupt file.		X				•	•	•

I/O Errors

ERROR CODES: The characters xx in the following codes represent the slot number, in hexadecimal. If the exact slot cannot be determined, the characters xx become 1F.

RECOVERABLE I/O FAULTS (SLC 5/02, SLC 5/03, SLC 5/04, and SLC 5/05 processors only): Many I/O faults are recoverable. To recover, you must disable the specified slot, xx, in the user fault routine. If you do not disable slot xx, the processor will fault at the end of the scan.

Note: *An I/O card that is severely damaged may cause the processor to indicate that an error exists in slot 1 even though the damaged card is installed in a slot other than 1.*

SLOT NUMBERS (xx) IN HEXADECIMAL

Slot	xx	Slot	xx	Slot	xx	Slot	xx
0	00	8	08	16	10	24	18
1	01	9	09	17	11	25	19
2	02	10	0A	18	12	26	1A
** 3	03	11	0B	19	13	27	1B
4	04	12	0C	20	14	28	1C
5	05	13	0D	21	15	29	1D
6	06	14	0E	22	16	30	1E
7	07	15	0F	23	17	*	1F

* This value indicates that the slot was not found (SLC 5/01, SLC 5/02, SLC 5/03, SLC 5/04, and SLC 5/05 processors).

** This value indicates that the slot was not found (500 fixed controller).

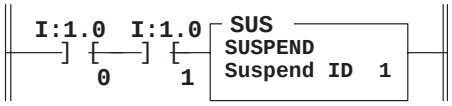
Address	Error Code (Hex)	User Program Instruction Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	0030	Attempt was made to jump to one too many nested subroutine files. This code can also mean that a program has potentially recursive routines.		X		•	•	•	•	•
	0031	An unsupported instruction reference was detected.		X		•	•	•	•	•
	0032	A sequencer length/position parameter points past the end of a data file.			X	•	•	•	•	•
	0033	The length of LFU, LFL, FFU, FFL, BSL, or BSR instruction points past the end of a data file.			X	•	•	•	•	•
	0034	A negative value for a timer accumulator or preset value was detected.			X	•	•	•	•	•
		Fixed processors with 24 VDC inputs only: A negative or zero HSC preset was detected in a HSC instruction.			X	•				
	0035	TND, SVC, or REF instruction is called within an interrupting or user fault routine.		X			•	•	•	•
	0036	An invalid value is being used for a PID instruction parameter.			X		•	•	•	•
	0038	A RET instruction was detected in a non-subroutine file.	X			•	•	•	•	•
	xx39	Invalid string length was detected in a string file. (xx = data file number)			X			•	•	•

Address	Error Code (Hex)	User Program Instruction Errors	Fault Classification			Processor				
			Non-User	User		Fixed 5/01	5/02	5/03	5/04	5/05
				Non-Recov	Recov					
	xx50	A chassis data error is detected. (xx = slot number)			X	•	•	•	•	•
	xx51	A "stuck" runtime error is detected on an I/O module. (xx = 31)		X		•	•	•	•	•
	xx52	A module required for the user program is detected as missing or removed. (xx = slot number)			X	•	•	•	•	•

Address	Error Code (Hex)	I/O Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	**xx53**	When going-to-run, a user program declares a slot as unused, and that slot is detected as having an I/O module inserted. This can also mean that an I/O module has reset itself. (xx = slot number)			X	•	•	•	•	•
		An attempt to enter the run or test mode was made with an empty chassis.			X			•	•	•
	xx54	A module required for the user program is detected as being the wrong type. (xx = slot number)			X	•	•	•	•	•
	xx55	A discrete I/O module required for the user program is detected as having the wrong I/O count. This code can also mean that a specialty card driver is incorrect. (xx = slot number)			X	•	•	•	•	•
	xx56	The chassis configuration specified in the user program is detected as being incorrect.	X			•	•	•	•	•
	xx57	A specialty I/O module has not responded to a lock shared memory command within the required time limit. (xx = slot number)			X	•	•	•	•	•
	xx58	A specialty I/O module has generated a generic fault. The card fault bit is set (1) in the module's status byte. (xx = slot number)		X		•	•	•	•	•

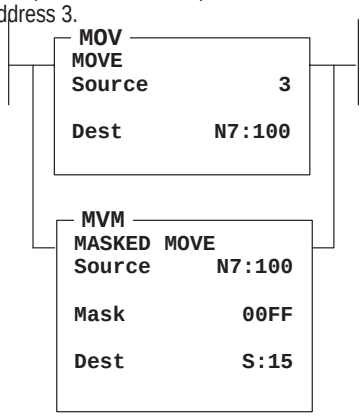
Address	Error Code (Hex)	I/O Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	**xx59**	A specialty I/O module has not responded to a command as being completed within the required time limit. (xx = slot number)			X	•	•	•	•	•
	xx5A	Hardware interrupt problem. (xx = slot number)		X			•	•	•	•
	xx5B	G file configuration error – user program G file size exceeds capacity of the module. (xx = slot number)			X		•	•	•	•
	xx5C	M0–M1 file configuration error – user program M0–M1 file size exceeds capacity of the module. (xx = slot number)			X		•	•	•	•
	xx5D	Interrupt service requested is not supported by the processor. (xx = slot number)			X		•	•	•	•
	xx5E	Processor I/O driver (software) error. (xx = slot number)			X		•	•	•	•
	xx60 to xx6F	Identifies an I/O module specific recoverable major error. Refer to the user manual supplied with the specialty module. (xx = slot number)			X		•	•	•	•
	xx70 to xx7F	Identifies an I/O module specific non-recoverable major error. Refer to the user manual supplied with the specialty module. (xx = slot number)		X			•	•	•	•

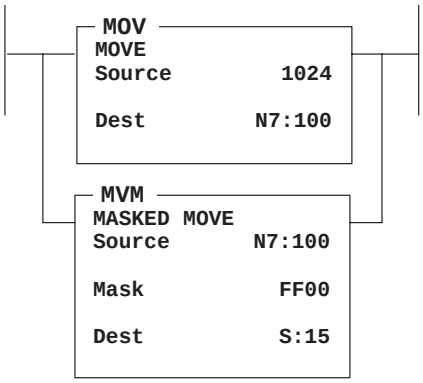
Address	Error Code (Hex)	I/O Errors	Fault Classification			Processor				
				User						
			Non-User	Non-Recov	Recov	Fixed 5/01	5/02	5/03	5/04	5/05
S:6 continued	xx80 to xx8F	Identifies a specialty I/O module specific major error. Refer to the user manual supplied with the specialty module. (xx = slot number)	X					•	•	•
	xx90	Interrupt problem on disabled slot.		X			•	•	•	•
	xx91	A disabled slot has faulted.		X			•	•	•	•
	xx92	An invalid or non-existent module interrupt subroutine (ISR) file.		X			•	•	•	•
	xx93	Unsupported I/O module specific major error.		X			•	•	•	•
	xx94	In the REM Run or REM Test mode, a module has been detected as being inserted under power. This can also mean that an I/O module has reset itself. (xx = slot number)		X			•	•	•	•
	0xYYA0	Indicates a communication channel hardware fault has occurred. With the SLC 5/05 only, the Ethernet channel (channel 1) may generate this fault, so the only possible value for this fault is 0x01A0. The fault may be cleared via a write to the System Status File, but Ethernet communications will be disabled until a power cycle is performed. Word 15 of the System Status File provides a specific fault code for the Ethernet Daughterboard when user fault code 0x01A0 is generated. (YY = channel number)			X					•

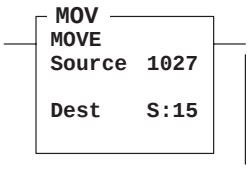
Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:7 and S:8	Status	**Suspend Code/Suspend File** When a non-zero value appears in S:7, it indicates that the SUS instruction identified by this value has been evaluated as true, and the Suspend Idle mode is in effect. This pinpoints the conditions in the application that caused the Suspend Idle mode. This value is not cleared by the processor. Word S:8 contains the program file number in which a true SUS instruction is located. This value is not cleared by the processor. Use the SUS instruction with startup troubleshooting, or as runtime diagnostics for detection of system errors. **Application example:** You believe that limit switches connected to I:1/0 and I:1/1 cannot be energized at the same time, yet your application program acts as if they can be. To determine if you have a limit switch problem or a ladder logic problem, add the following rung to your program:  `I:1.0 0` `I:1.0 1` `SUS SUSPEND Suspend ID 1` If your program enters the SUS Idle mode for code 1 when you run the program, you have a limit switch control problem; if the SUS Idle mode for code 1 does not occur, you have a ladder logic problem.	•	•	•	•	•
S:9	Status	**Ethernet Daughter Board Firmware Series (Channel 1 – SLC 5/05 processors)** A value of 1 corresponds to Series 1, for example.					•
S:10	Status	**Ethernet Daughter Board Firmware Revision (Channel 1 – SLC 5/05 processors)** A value of 12 corresponds to Revision 12, for example.					•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:9 and S:10 continued	Status	**Active Nodes (Channel 1–SLC 5/03 processors)** These two words are bit mapped to represent the 32 possible nodes on a DH-485 link. S:9/0 through S:10/15 represent node addresses 0–31. These bits are set by the processor when a node exists on the DH-485 link that your processor is connected to. The bits are cleared when a node is not present on the link.	•	•	•		
S:11 and S:12	Dynamic Config	**I/O Slot Enables** These two words are bit mapped to represent the 30 possible I/O slots in an SLC 500 system. S:11/0 represents I/O slot 0 for fixed I/O systems. (Slot 0 is used for the CPU in modular systems.) S:11/1 through S:12/14 represent I/O slots 1–30. S:12/15 is unused. When a bit is set (default condition), it allows the I/O module contained in the referenced slot to be updated in the I/O scan of the processor operating cycle. When you clear a bit, it causes the I/O module in the referenced slot to be ignored. That is, an I/O slot enable value of 0 causes the input image data of an input module to freeze at its last value. Also, the outputs of an output module will freeze at their last values, regardless of values contained in the output image. Outputs remain frozen until: • either power is removed, • the REM Run mode is exited, or • a major fault occurs. At that time the outputs are zeroed, until the slot is again enabled (set). Disabled slots do not have to match the user program configuration. ⚠ **Make certain that you have thoroughly examined the effects of disabling (clearing) a slot enable bit before doing so in your application.**	•	•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:11 and S:12 continued		**Note:** *The SLC 5/02 and higher processors inform each specialty I/O module that has been disabled/enabled. Some I/O modules may perform other actions when disabled or re-enabled. Refer to the user information supplied with the specialty I/O module for possible differences from the above descriptions.*		•	•	•	•
		⚠ The DII instruction ignores the slot enable/disable status. Do not run the DII on a faulted slot. If you apply the DII on a disabled slot, the interrupt will occur. However, the input image will not reflect the present state of the card. **This bit is applied upon detection of a DII Reconfigure bit, each DII ISR exit, and at each end of scan (END, TND, or REF).**			•	•	•
S:13 and S:14	**Status and Dynamic Config**	**Math Register** Use this double register to produce 32-bit signed divide and multiply operations, precision divide or double divide operations, and 5-digit BCD conversions. These two words are used in conjunction with the MUL, DIV, DDV, FRD, and TOD math instructions. The math register value is assessed upon execution of the instruction and remains valid until the next MUL, DIV, DDV, FRD, or TOD instruction is executed in the user program. An explanation of how the math register operates is included with the instruction definitions. If you store 32-bit signed data values (example on page 3–8), you must manage this data type without the aid of an assigned 32-bit data type. For example, combine B10:0 and B10:1 to create a 32-bit signed data value. We recommend that you keep all 32-bit signed data in a unique data file and that you start all 32-bit values on an even or odd word boundary for ease of application and viewing. Also, we recommend that you design, document, and view the contents of 32-bit signed data in either the hexadecimal or binary radix. See chapter 3 for more information on how the math register is effected by each instruction.	•	•	•	•	•

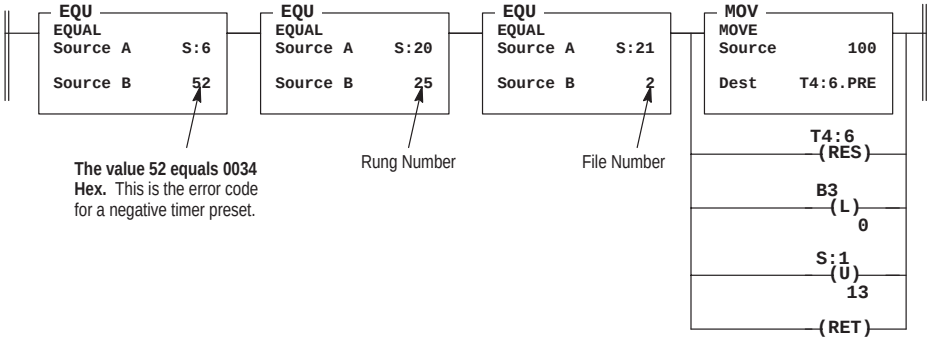
Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:13 and S:14 continued		When an STI, I/O Slot, or Fault Routine interrupts normal execution of your program, the original value of the math register is restored when execution resumes. Note that S:13 and S:14 are not used when the source or destination is defined as floating point data.		•	•	•	•
		When a DII interrupts normal execution of your program, the original value of the math register is restored when execution resumes.			•	•	•
S:15L	**Static Config**	**Node Address** This byte value contains the node address of your processor on the DH-485 or DH+ link. Each device on the DH-485 link must have a unique address between the decimal values 0–31. Each device on the DH+ link must have a unique address between the decimal values 0–63. To change a processor node address, write a value between 1–31 for DH-485 and 0–63 for DH+ using either the Data Monitor or node function of your programmer, then cycle power to the processor. The default node address of a processor is 1. The default DH-485 node address of APS and the HHT programmer is 0. To provide runtime protection from inadvertent data monitor alteration of your selection, program this value using an unconditional MVM instruction. Use the MOV instruction in place of MVM if you also wish to protect the baud rate. The following example shows runtime protection of node address 3. 	•	•	•	•	

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:15H continued	**Static Config**	When a configure channel command is received for channel 1, the node address is overwritten with the value contained in your channel configuration.			•	•	
		Baud Rate This byte value contains a code used to select the baud rate of the processor on the DH-485 or DH+ link. SLC 5/01 and fixed processors provide a baud rate of 19.2K or 9.6K only. SLC 5/02 and SLC 5/03 processors provide a baud rate of 19.2K, 9.6K, 2.4K, or 1.2K. SLC 5/04 processors provide a baud rate of 57.6K, 115.2K, and 230.4K. To change the baud rate from the default values of 19.2K or 57.6K, use either the Data Monitor or baud function of your programmer. The processor uses code 1 for 1.2K, code 2 for 2.4K, code 3 for 9.6K, code 4 for 19.2K, code 11 for 57.6K, code 12 for 115.2K, and code 13 for 230.4K baud. Example showing runtime protection of baud rate 19.2K (code 4):  S:15H equal to 4 = 1024 decimal = 0400 hex = 0000 0100 0000 0000 binary	•	•	•	•	

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:15 continued	Static Config	Example showing runtime protection for both baud rate 19200 (code 4) and node address 3:  S:15H equal to 4 and S:15L equal to 3 = 1027 decimal = 0403 hex = 0000 0100 0000 0011 binary					
		When a configure channel command is received for channel 1, the baud rate is overwritten with the value contained in your channel configuration.			•	•	
	Static Config	**Ethernet Daughter Board Fault Code** This word value contains the fault code when User Fault Code 0x01A0 occurs.					•
S:16 and S:17	Status	**Test Single Step – Start Step On – Rung/File** These registers indicate the executable rung (word S:16) and file (word S:17) number that the processor executes next when operating in the Test Single Step mode. To enable this feature, you must select the Test Single Step option at the time you save your program. These values are updated upon completion of every rung. Refer to word S:2/4 for more information. Your programming device interrogates this value when providing “start step on file x, rung y” status line information. There is no known use for this feature when addressed by your ladder program.		•	•	•	•
		This feature is built into the SLC 5/03 and higher processors. Selection is not required.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:18 and S:19	Status and Dynamic Config	**Test Single Step – Breakpoint – Rung/File** These registers indicate the executable rung (word S:18) and file (word S:19) number that the processor should stop in front of when executing in the Test Single Step mode. To enable this feature, you must select the Test Single Step option at the time you save your program. If both the rung and file number are 0, the processor steps to the next rung only; otherwise the processor continues until it finds a rung/file equaling the S:18/S:19 value. The processor stops, then clears S:18 and S:19 when it finds a match, while remaining in the Test Single Step mode. The processor operates indefinitely if it cannot find the end rung/file that you have entered. It operates until it finds a match, receives a mode change, or powers down. See S:2/4. Your programming device interrogates this value when providing “end step before file x, rung y” status line information. Your programming device also writes this value when prompting you for “set end rung.” There is no known use for this feature when addressed by your ladder program.		•	•	•	•
		This feature is built into the SLC 5/03 and higher processors. Selection is not required.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:20 and S:21	Status	**Test – Fault/Powerdown – Rung/File** These registers indicate the executable rung (word S:20) and file (word S:21) number that the processor last executed before a major error or powerdown occurred. To enable this feature, you must select the Test Single Step option at the time you save your program. You can use these registers to pinpoint the execution point of the processor at the last powerdown or fault routine entry. This function is also active in the REM Run mode. See S:2/4. **Application example:** Suppose your program contains several TON instructions. TON T4:6 in file 2, rung 25 occasionally obtains a negative preset. Recovery from the negative preset fault is possible by placing the preset at 100 and resetting the timer. Place the following rung in your fault routine to accomplish this. Bit B3/0 is latched as evidence that an application recovery has been initiated.		•	•	•	•
		This feature is built into the SLC 5/03 and higher processors. Selection is not required.			•	•	•



Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:22	Status	**Maximum Observed Scan Time** This word indicates the maximum observed interval between consecutive scans. Consecutive scans are defined as intervals between file 2/rung 0 and the END, TND, or the REF instruction. This value indicates, in 10 ms increments, the time elapsed in the longest program cycle of the processor. The processor compares each last scan value to the value contained in S:22. If the processor determines that the last scan value is larger than the value stored at S:22, the last scan value is written to S:22. Resolution of the maximum observed scan time value is +0 to –10 ms. For example, the value 9 indicates that 80–90 ms was observed as the longest program cycle. Interrogate this value using the Data Monitor function if you need to determine or verify the longest scan time of your program. **Note:** *The I/O scan, processor overhead, and communication servicing is* not *included in this measurement.*		•	•	•	•
		The Scan Time Selection Bit (S:33/13) determines the timebase used for average and maximum Scan Times. When clear, operation is as described above. When set, the timebase is expressed in 1 ms increments (instead of 10 ms increments). When S:33/13 is set, resolution of the maximum observed scan time value is +0 to –1 ms. For example, the value 9 indicates that 8 to 9 ms was observed as the largest program cycle.			•	•	•
S:23	Status	**Average Scan Time** This word indicates a weighted running average time. The value indicates, in 10 ms increments, the time elapsed in the average program cycle of the processor. For every Scan *t*: $\text{Avg} = \frac{(\text{Avg} * 7) + \text{Scan}_t}{8}$ Resolution of the average scan time value is +0 to –10 ms. For example, the value 2 indicates that 10 to 20 ms was calculated as the average program cycle. **Note:** *The I/O scan, processor overhead, and communication servicing is* not *included in this measurement.*		•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:23 continued		The Scan Time Selection bit S:33/13 determines the timebase used for average Scan time. When clear, operation is as described above. When set, the timebase is expressed in 1 ms increments (instead of 10 ms increments). When S:33/13 is set, resolution of the average scan time value is +0 to –1 ms. For example: the value 2 indicates that 1 to 2 ms was calculated as the average program cycle.			•	•	•
S:24	**Dynamic Config**	**Index Register** This word indicates the element offset used in indexed addressing. When an STI, I/O Slot, or Fault Routine interrupts normal execution of your program, the original value of this register is restored when execution resumes.		•	•	•	•
		When a DII interrupts normal execution of your program, the original value of this register is restored when execution resumes.			•	•	•
S:25 and S:26	**Status**	**I/O Interrupt Pending** These two words are bit-mapped to the 30 I/O slots. Bits S:25/1 through S:26/14 refer to slots 1–30. Bits S:25/0 and S:26/15 are reserved. The pending bit associated with an interrupting slot is set when the corresponding I/O Slot Interrupt Enable bit is clear at the time of an interrupt request. It is cleared when the corresponding I/O Event Interrupt Enable bit is set, or when an associated RPI instruction is executed. The pending bit for an executing I/O interrupt subroutine remains clear when the ISR is interrupted by an STI or fault routine. Likewise, the pending bit remains clear if interrupt service is requested at the time that a higher or equal priority interrupt is executing (fault routine, STI, or other ISR). I/O interrupts are discussed in chapter 12 of this manual.		•	•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:25 and S:26 continued		The pending bit associated with an interrupting slot is set when the corresponding I/O Slot Interrupt Enable bit is clear at the time of an interrupt request. It is cleared when the corresponding I/O Event Interrupt Enable bit is set, or when an associated RPI instruction is executed. The pending bit is always set when interrupt service is requested and the processor is executing an interrupt of equal or higher priority. Interrupt priority does not affect the setting of these bits. For example, while executing an STI subroutine, slot 6 requests an I/O Event Interrupt. The STI executes to completion; however, slot 6 pending bit (S:25/6) becomes set within execution of the STI. Examine the state of these bits within your interrupt subroutines if your application requires this information.			•	•	•
S:27 and S:28	**Status**	**I/O Interrupt Enabled** These two words are bit-mapped to the 30 I/O slots. Bits S:27/1 through S:28/14 refer to slots 1–30. Bits S:27/0 and S:28/15 are reserved. The default value of each bit is 1 (set). The enable bit associated with an interrupting slot must be set when the interrupt occurs to allow the corresponding ISR to execute. Otherwise, the ISR does not execute and the associated I/O slot interrupt pending bit becomes set. Changes made to these bits using the Data Monitor function or ladder instructions other than IID or IIE take affect at the next end of scan. I/O interrupts are discussed in chapter 12 of this manual.		•	•	•	•
	Dynamic Config	These bits may be set/reset by the user program, comms., or with the IIE or IID instruction. Changes made to these bits using a programming terminal's Data Monitor function or any ladder instruction take effect immediately.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:29	Dynamic Config	**User Fault Routine File Number** You enter a program file number (3–255) to be used in all recoverable and non-recoverable major errors. Program the ladder logic of your fault routine in the file you have specified. Write a zero value to disable the fault routine. To provide protection from inadvertent Data Monitor alteration of your selection, program an unconditional MOV instruction containing the program file number of your fault routine to S:29, or program a CLR instruction at S:29 to prevent fault routine operation. The fault routine is discussed in chapter 12 of this manual.		•	•	•	•
S:30	Dynamic Config	**Selectable Timed Interrupt – Setpoint** You enter the timebase, in tens of milliseconds, to be used in the selectable timed interrupt. Your STI routine executes per the value you enter. Write a zero value to disable the STI. To provide protection from inadvertent Data Monitor alteration of your selection, program an unconditional MOV instruction containing the setpoint value of your STI to S:30, or program a CLR instruction at S:30 to prevent STI operation. If the STI is initiated while in the REM Run mode by loading the status registers, the interrupt starts timing from the end of the program scan in which the status registers were loaded. Selectable timed interrupts are discussed on page 12–7 of this manual.		•	•	•	•
		The STI Setpoint timebase can be either 10 ms or 1 ms depending on the value of the STI Setpoint Selection bit S:2/10. When clear, operation is as described above. When set, the timebase is expressed in 1 ms increments. The STE enables and STD disables the STI instruction.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:31	Dynamic Config	**Selectable Timed Interrupt – File Number** You enter a program file number (3–255) to be used as the selectable timed interrupt subroutine. Write a zero value to disable the STI. To provide protection from inadvertent Data Monitor alteration of your selection, program an unconditional MOV instruction containing the file number value of your STI to S:31, or program a CLR instruction at S:31 to prevent STI operation. Selectable timed interrupts are discussed on page 12–7 of this manual.		•	•	•	•
S:32	Status	**I/O Interrupt Executing** This word indicates the slot number of the specialty I/O module that generated the currently executing ISR. This value is cleared upon completion of the ISR, REM Run mode entry, or upon power-up. You can interrogate this word inside of your STI subroutine or fault routine if you wish to know if these higher priority interrupts have interrupted an executing ISR. You may also use this value to discern interrupt slot identity when multiplexing two or more specialty I/O module interrupts to the same ISR. I/O interrupts are discussed on page 12–7 of this manual.		•	•	•	•
		You can interrogate this word inside your DII subroutine if you wish to know if these higher priority interrupts have interrupted an executing ISR. You may also use this value to discern interrupt slot identity when multiplexing two or more specialty I/O module interrupts to the same ISR.			•	•	•
S:33/0	Status	**Incoming Command Pending (Channel 0)** This bit becomes set when the processor determines that another node on the channel 0 network has requested information or supplied a command to it. This bit can be set at any time. This bit is cleared when the processor services the request (or command). Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:33/1	Status	**Message Reply Pending (Channel 0)** This bit becomes set when another node on the channel 0 network has supplied the information that you requested in the MSG instruction of your processor. This bit is cleared when the processor stores the information and updates your MSG instruction. Use this bit as a condition of an SVC instruction to enhance the communication capability of your processor.			•	•	•
S:33/2	Status	**Outgoing Message Command Pending (Channel 0)** This bit is set when one or more channel 0 messages in your program are enabled and waiting, but no message is being transmitted at the time. As soon as transmission of a message begins, the bit is cleared. After transmission, the bit is set again if there are further messages waiting, or it remains cleared if there are no further messages waiting.			•	•	•
S:33/3	Status	**Selection Status (Channel 0)** When set, this bit indicates that the channel 0 communication port is in the System mode (DF1 mode). When reset, this bit indicates that channel 0 is in the User mode (ASCII mode). Use your programming devices channel configuration utility to change this selection.			•	•	•
S:33/4	Status	**Communications Active (Channel 0)** *DH-485 protocol only* – This bit is set by the processor when at least one other node is active on channel 0 DH-485 network. Otherwise the bit remains cleared.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:33/5	Dynamic Config	**Communications Servicing Selection (Channel 0)** When set, only one channel 0 communication request/ command will be serviced per END, TND, REF, or SVC instruction. When clear, all serviceable incoming or outgoing communication requests/commands will be serviced per END, TND, REF, or SVC instruction. One communication request/command consists of either a channel 0 Incoming Command, channel 0 Message Reply, or channel 0 Outgoing Message Command. Refer to Words S:33/0, S:33/1, S:33/2, and S:33/6 for more information. **Note:** *When clear, your communication throughput will increase. Your scan time will also increase if several communication commands/requests are received in the same scan.* To program this feature, use the Data Monitor function to set and clear this bit. To provide protection from inadvertent data monitor alteration of your selection, program an unconditional OTL instruction at address S:33/5 to ensure one request/command operation, or an unconditional OTU instruction at address S:33/5 to ensure multiple request/command operation. Alternately, your program may change the state of this bit using ladder logic if your application requires dynamic selection of this function.			•	•	•
S:33/6	Dynamic Config	**Message Servicing Selection (Channel 0)** This bit is only valid when the channel 0 Comms Servicing Selection (S:33/5) is clear (which selects service all commands). When S:33/6 is set and S:33/5 is clear, all outgoing channel 0 MSG instructions will be serviced per END, TND, SVC, or REF instruction. Otherwise, only one outgoing channel 0 MSG command or reply will be serviced per END, TND, SVC, or REF instruction.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:33/7	Dynamic Config	**Message Servicing Selection (Channel 1)** This bit is only valid when the channel 1 Comms Servicing Selection bit (S:2/15) is clear (which selects service all commands). When S:33/7 is set and S:2/15 is clear, all outgoing channel 1 MSG instructions are serviced per END, TND, SVC, or REF instruction. Otherwise, only one outgoing channel 1 MSG command or reply is serviced per END, TND, SVC, or REF instruction.			•	•	•
S:33/8	Static Config	**Interrupt Latency Control Bit** When set, interrupt latency occurs for user interrupts (DII, STI, and I/O Event). This means that when an interrupt occurs, you are guaranteed to be at rung 0 of your interrupt subroutine within the stated interrupt latency period (assuming no interrupt of equal or higher priority is executing). You must select this at the time you save your program. When clear, user interrupts may only interrupt the processor at predefined points of execution in the user program cycle. Interrupt latency is then defined as the longest period of time that can occur between any two predefined points. When S:33/8 is clear, you must analyze each user program. The bit is clear by default. The following points are the only points in which user interrupt subroutines are allowed to execute when S:33/8 is clear: • at the start of each rung • following the servicing of communication • between slots when updating the input or output image, or any specialty I/O card			•	•	•
S:33/9	Status	**Scan Toggle Bit** This bit is cleared upon entry into the RUN mode. This bit changes state each and every execution of an END, TND, or REF instruction. Use this bit in your user program for applications such as multiplexing subroutine execution.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05

S:33/10	Dynamic Config	**Discrete Input Interrupt Reconfiguration Bit**
		Set this bit with your user program or programming terminal to cause the DII function to reconfigure itself at the next interrupt occurrence or end of each scan (END, TND, or REF). This bit is applied upon a DII ISR, fault routine, STI ISR, or Event ISR exit.
		The following occurs when the DII is reconfigured:
		1. The DII Accumulator is cleared (S:52).
		2. DII parameters located in words S:46 through S:50 are applied.
		3. The DII reconfigure bit is cleared by the processor.
		For example, use the following ladder structure to cause a DII reconfiguration from your main ladder file each time input 0 is cycled on.
		`I:1/0 B3/0 S:33/10`
		`]  [ [OSR] (L)`
		Use the following ladder structure to cause a DII reconfiguration from an event based subroutine. The subroutine is only executed once, each time the DII reconfiguration is possible.
		`I:1/0 S:33/10`
		`]  [ (L)`
					•	•	•

S:33/11 and S:33/12	Status	**Online Edit Status** These two bits represent the four possible Online Edit states: **Bit 12 / Bit 11 / Online Edit Status** 0 / 0 / No online edits exist 0 / 1 / Online edits are disabled 1 / 1 / Testing online edits 1 / 0 / not used Examine the state of these bits with your user program to count the number of online edit sessions, flag an alarm, or place your application in a special state designed for online edit sessions.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:33/13	Static Config	**Scan Time Timebase Selection** This bit determines the timebase used to average the Scan time (S:23) and the maximum Scan Time (S:22). When clear, the value contained in the average and maximum scan times represent the number of 10 ms increments that have occurred. When set, the value contained in the average and maximum scan times represent the number of 1 ms increments that have occurred. This value is clear by default (10 ms timebase).			•	•	•
S:33/14	Dynamic Config	**DTR Control Bit (Channel 0)** This bit is used to enable DTR dialing. When clear, the channel 0 DTR signal (pin 4) is directly controlled by the standard communication driver. When set, you can perform DTR dialing by writing to S:33/15, DTR Force Bit. Bit S:33/14 is examined and applied at each end of scan (END, TND, or REF). When in Program, Suspend, or Fault mode, DTR is enabled and remains enabled until an auto–disconnect sequence is detected by the communication driver. An auto–disconnect occurs if the communication driver detects that channel 0 CD signal (pin 1) has been absent for more than 10 seconds or if the channel 0 DSR signal (pin 6) has been disabled. Refer to S:5/14 Channel 0 Modem Lost bit for more information. During an auto–disconnect, the standard communication driver keeps the DTR disabled until either the channel 0 DSR signal is enabled, or 5 seconds elapse. **Note:** *When channel 0 is configured for DH485, S:33/14 must be clear for proper operation.*			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:33/15	Dynamic Config	**DTR Force Bit (Channel 0)** This bit is used to force the DTR pin high or low. When S:33/14 is set, the channel 0 DTR signal (pin 4) is applied at each end of scan (END, TND, or REF) using the state of S:33/15. When S:33/14 is clear, this bit has no effect on DTR. When S:33/15 is set, DTR is forced high. When clear (default), DTR is forced low. When in the REM Test or REM Run mode, this bit is only applied at end of scan (END, TND, or REF). When in Program, Suspend, or Fault mode (or on power up), DTR is set unless the communication driver is performing an auto–disconnect.			•	•	•
S:34/0	Static Config	**DH+ to DH-485 Passthru Disabled Bit** This bit provides the capability to pass received packets between channels. When set, the processor does not support passthru. When reset, the processor allows packets to be passed from one channel to the other. Channel 0 (RS-232) must be configured for DH-485 protocol. Only packets that contain the Internet network layer and whose Destination Link ID equals that specified for the opposite channel will be passed. The default is reset. The default Link ID for channel 0 is one. The default Link ID for channel 1 is two.				•	
S:34/1	Static Config	**DH+ Active Node Table Enable Bit** This bit enables processing of the DH+ active node table. When set, the DH+ active node table is processed. When clear, the DH+ active node table is not processed. The default is clear. This bit is evaluated upon each entry into the REM Run mode. Note that the processor updates individual status words S:83 to S:86.				•	

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:34/2	Dynamic Config	**Floating Point Math Flag Disable Bit** This bit disables the processing of math flags when using floating point math (F8:). The math flags effected are Overflow (S:0/1), Zero (S:0/2), Sign (S:0/3), and the Minor Error Overflow Trap bit (S:5/0). When the bit is clear, the math flags are processed. When the bit is set, the math flags are cleared except for the Minor Error Overflow Trap bit which remains in its last state. The Carry Flag (S:0/0) is reserved for internal use during all floating point operations. The default is clear. Instructions effected by floating point include ADD, SUB, MUL, DIV, NEG, SQR, and MOV. Setting this bit reduces the execution times for the above instructions. This bit is evaluated when each instruction is executed.			•	•	
S:34/2	Dynamic Config	**Update Math Status Bits**					•
S:34/3	Dynamic Config	**Global Status Word Transmit Enable Bit (SLC 5/04 OS401 only)** When this bit is set, the Global Status Word at S:99 is transmitted with every DH+ token pass. When clear, the token is passed without the Global Status Word.				•	
S:34/4	Dynamic Config	**Global Status Word Receive Enable Bit (SLC 5/04 OS401 only)** When this bit is set, the processor collects the Global Status Word being transmitted by other devices on the DH+ network and stores them in the Global Status File (S:100–S:163). When clear, the processor ignores the Global Status information from other devices on the network.				•	

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:34/5	Dynamic Config	**DF1 to DH+ Passthru Enabled Bit (SLC 5/04 OS401 only)** When this bit is set, passthru operation is enabled between Channel 0 and Channel 1. Channel 0 must be configured for DF1 full-duplex protocol. Only DH+ packets that contain the internet network layer and whose Destination Link ID equals that configured for channel 0 will be passed from channel 1 to channel 0. Only DF1 packets whose destination address (DST) is a valid DH+ address (0–63), and does not equal the DH+ address of this SLC 5/04 processor, will be passed from channel 1 to channel 0.				•	
S:35	Status	**Last 1 ms Scan Time** The value of this word tells you how much time elapsed in a program cycle. A program cycle includes the ladder program, housekeeping, I/O scan, and servicing of the communication port. This word value is only updated by the processor once each scan, immediately preceding the execution of rung 0, file 2 (or upon return of a REF instruction).			•	•	•
S:36/0 to S:36/7	NA	**Reserved**			•	•	•
S:36/8	Status	**DII Lost** This bit is set anytime a DII interrupt occurs while the DII Pending bit (S:2/11) is also set. When set, you are notified that a DII interrupt has been lost. For example, the interrupt is lost because a previous interrupt was already pending and waiting execution. Examine this bit in your user program and take appropriate action if your application cannot tolerate this condition. Then clear this bit with your user program to prepare for the next possible occurrence of this error.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:36/9	Status	**STI Lost** This bit is set anytime an STI interrupt occurs while the STI Pending bit (S:2/0) is also set. When set, you are notified that a STI interrupt has been lost. For example, the interrupt is lost because a previous interrupt was already pending and waiting execution. Examine this bit in your user program and take appropriate action if your application cannot tolerate this condition. Then clear this bit with your user program to prepare for the next possible occurrence of this error.			•	•	•
S:36/10	Status	**Memory Module Data File Overwrite Protection** Use this bit to determine the validity of retentive data following a memory module transfer. This bit is always set when a memory module to processor transfer occurs with Data File Overwrite Protection selected and protected files are overwritten. Protected files are overwritten anytime a memory module program does not match the processor program at the time of the transfer. This bit is not cleared by the processor.			•	•	•
S:36/11 to S:36/15	NA	**Reserved for additional minor errors.**			•	•	•
S:37	Dynamic Config	**Clock/Calendar Year** This value contains the year value of the clock/calendar. Valid range is 0–65535. To disable the clock/calendar, write zeros to all clock/calendar words (S:37 to S:42).			•	•	•
S:38	Dynamic Config	**Clock/Calendar Month** This value contains the month value of the clock/ calendar. Valid range is 1–12. To disable the clock/calendar, write zeros to all clock or calendar words (S:37 to S:41). January equals the value of 1.			•	•	•
S:39	Dynamic Config	**Clock/Calendar Day** This value contains the day value of the clock/calendar. Valid range is 1–31. To disable the clock/calendar, write zeros to all clock or calendar words (S:37 to S:41). The first day of the month equals the value of 1. See status word S:53 for Day-of-Week.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:40	Dynamic Config	**Clock/Calendar Hours** This value contains the hour value of the clock/calendar. Valid range is 0–23. To disable the clock/calendar, write zeros to all clock or calendar words (S:37 to S:41). 0000 hundred hours equals the value of 0.			•	•	•
S:41	Dynamic Config	**Clock/Calendar Minutes** This value contains the minute value of the clock/calendar. Valid range is 0–59. To disable the clock/calendar, write zeros to all clock or calendar words (S:37 to S:41).			•	•	•
S:42	Dynamic Config	**Clock/Calendar Seconds** This value contains the seconds value of the clock/calendar. Valid range is 0–59. To disable the clock/calendar, write zeros to all clock or calendar words (S:37 to S:41).			•	•	•
S:43 S:44 S:45	Status	**Selectable Timed Interrupt – 10 μs Timer** **I/O Event Interrupt – 10 μs Timer** **Discrete Input Interrupt – 10 μs Timer** This 16-bit value is "free running" and is used to measure the amount of time that expires between consecutive interrupt subroutine executions (in increments of 10 μs). This value is updated upon each entry into the interrupt subroutine. The 10 μs timer dictates that the maximum amount of time that can expire between any two interrupts and still result in a valid time measurement is 0.32767 seconds. (16-bit signed $\times$ 10 μs = 32767 $\times$ .00001 = 0.32767 seconds) The 10 μs timer is common to the STI interrupt, the Event I/O interrupt, and the DII interrupt.			•	•	•
S:46	Dynamic Config	**Discrete Input Interrupt – File Number** You enter a program file number (3–255) to be used as the discrete input interrupt subroutine. Write a zero value to disable the function. This value is applied upon detection of a DII Reconfigure bit, each DII ISR exit, and each end of scan (END, TND, or REF). To provide protection from inadvertent data monitor alteration of your selection, program an unconditional MOV instruction containing the file number value of your DII to S:46 or program a CLR instruction at S:46 to prevent DII operation.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:47	Dynamic Config	**Discrete Input Interrupt – Slot Number** You enter the slot number (1–30) that contains the Discrete I/O module to be used as the discrete input interrupt slot. The processor will fault if the slot is empty or contains a non–discrete I/O module. For example, an analog module causes a processor fault to occur. This bit is applied upon detection of the DII Reconfigure bit. This value is only applied upon execution of the DII reconfiguration function (setting bit S:33/10 or upon REM Run mode entry with the DII Enable bit S:2/12 set). To provide protection from inadvertent data monitor alteration of your selection, program an unconditional MOV instruction containing the slot number value of your DII to S:47.			•	•	•
S:48	Dynamic Config	**Discrete Input Interrupt – Bit Mask** You enter a bit mapped value that corresponds to the bits that you wish to monitor on the discrete I/O module. Only bits 0 to 7 are used in the DII function. Setting a bit indicates that you wish to include the bit in the comparison of the discrete I/O module's bit transition to the DII Compare Value (S:49). Clearing a bit indicates that the transition state of that particular bit is a "don't care" bit. This value is applied upon detection of a DII Reconfigure bit, each DII ISR exit, and at each end of scan (END, TND, or REF). To provide protection from inadvertent data monitor alteration of your selection, program an unconditional MOV instruction containing the bit mask value of your DII to S:48.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:49	Dynamic Config	**Discrete Input Interrupt – Compare Value** You enter a bit mapped value that corresponds to the bit transitions that must occur in the discrete I/O card for a count or interrupt to occur. Only bits 0 to 7 are used in the DII function. Setting a bit indicates that the bit must transition from a 0 to a 1 to satisfy the compare condition for that bit. Clearing a bit indicates that the bit must transition from a 1 to a 0 in order to satisfy the compare condition for that bit. An interrupt or count will be generated upon the last bit transition of the compare value. This value is applied upon detection of a DII Reconfigure bit, each DII ISR exit, and each end of scan (END, TND, or REF). To provide protection from inadvertent data monitor alteration of your selection, program an unconditional MOV instruction containing the compare value of your DII to S:49.			•	•	•
S:50	Dynamic Config	**Discrete Input Interrupt – Preset** When this value is equal to 0 or 1, an interrupt is generated each time the comparison specified in words S:48 and S:49 is satisfied. When this value is between 2 and 32767, a count will occur each time the bit comparison is satisfied. An interrupt will be generated when the accumulator value reaches 1 or exceeds the preset value. This value is applied on detection of DII Reconfigure bit, each DII ISR exit, and at each end of scan (END, TND, or REF). To provide protection from inadvertent data monitor alteration of your selection, program an unconditional MOV instruction containing the preset value of your DII to S:50.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:51	Status	**Discrete Input Interrupt – Return Mask** The return mask is updated immediately preceding entry into the DII subroutine. This value contains the bit map of the bit transitions that caused the interrupt. The bit is set if it was included in the list of bit transitions that caused the interrupt, (specified to transition in the S:48 and S:49 comparisons). The bit is cleared if it was masked. This value is cleared by the processor upon exit of the DII subroutine. Use this value to validate the interrupt transitions. Or when dynamically reconfiguring (sequencing) the DII, you can use this value inside your DII's subroutine to help determine or validate its position in the sequence.			•	•	•
S:52	Status	**Discrete Input Interrupt – Accumulator** The DII accumulator contains the number of counts that have occurred (see S:50.) When a count occurs, and the accumulator is greater than or equal to the preset value, a DII interrupt is generated.			•	•	•
S:53L	Dynamic Config	**Day-of-Week** This value contains the day–of–week value of the clock/calendar. Valid range is 0–6 (Sunday=0). To disable the clock/calendar, write zeros to all clock and calendar words (S:37 to S:41).			•	•	•
S:53H	NA	**Reserved**			•	•	•
S:54	NA	**Reserved**			•	•	•
S:55	Status	**Last Discrete Input Interrupt Scan Time** This value indicates, in 1 ms increments, the amount of time elapsed by the most recent DII subroutine. The resolution of this value is +0 to –1 ms.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:56	Status	**Maximum Observed Discrete Input Scan Time** This value indicates, in 1 ms increments, the maximum amount of time elapsed by any single DII subroutine execution. The processor compares each last DII scan value (S:55) to the maximum DII scan value contained in S:56. If the processor determines that the last DII scan value is larger than the value stored at S:56, the last scan value (S:55) is written to S:56, thus becoming the new maximum DII scan time. The resolution of this value is +0 to –1 ms. Interrogate this value using a programming device Data Monitor function if you need to determine or verify the longest scan time of your program.			•	•	•
S:57	Status	**Operating System Catalog Number** Indicates the operating system catalog number. For example, the value of 300 indicates operating system -OS300, the value of 301 indicates -OS301.			•	•	•
S:58	Status	**Operating System Series** Indicates the operating system series. For example, the value of 0 indicates series A and the value of 1 indicates series B.			•	•	•
S:59	Status	**Operating System FRN** Indicates the operating system firmware release number. For example, the value of 1 indicates FRN1 and the value of 2 indicates FRN2.			•	•	•
S:60	Status	**Processor Catalog Number** Indicates the catalog number of the processor. For example, the value of 532 indicates -L532 and the value of 534 indicates -L534.			•	•	•
S:61	Status	**Processor Series** Indicates the processor series. For example, the value of 0 indicates series A and the value of 1 indicates series B.			•	•	•
S:62	Status	**Processor Revision** Indicates the processor revision. For example, the value of 1 indicates REV1 and the value of 2 indicates REV2.			•	•	•
S:63	Status	**User Program Type** Indicates the programming device that created the user program.			•	•	•

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:64	Status	**User Program Functionality Index** Indicates the level of functionality contained in a given program type.			•	•	•
S:65	Status	**User RAM Size** Applies to SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors. Indicates the size of NVRAM in instruction words. For example, the value 64 equals 64K instruction words of NVRAM.			•	•	•
S:66	Status	**Flash EEPROM Size** Indicates the size of operating system memory in 16 bit K words. For example, the value of 128 equals 128K words of memory.			•	•	•
S:67 to S:82	Status	**Channel 0 Active Node Table** Only used when channel 0 is configured for DF1 half-duplex master. These 16 words are bit-mapped to represent the 255 possible nodes in a DF1 half-duplex network. Bit S:67/0 represents node 0, bit S:82/14 represents node 254, and bit S:82/15 is not used since address 255 is reserved for broadcasts. These bits are set by the processor, which is configured for either of the two “standard” polling modes, when a node is polled (because it appears in either the normal or priority polling ranges) and it responds to the master poll. A node’s bit is cleared if no response is received after a master poll.			•	•	•
S:83 to S:86	Status	**Channel 1 Active Node Table** These 4 words are bit mapped to represent the 64 possible nodes on a DH+ link. S:83/0 through S:86/15 represent node addresses 0–63 (0–77 octal). These bits ares set by the processor when a node exists on the DH+ link that your processor is connected to. These bits are cleared when a node is not present on the link. Note that S:34/1 must be set for the above words to work.				•	
S:87 to S:96	NA	**Reserved**				•	•
S:97 to S:98	NA	**Reserved (Applies to SLC 5/04 OS 401 processors.)**				•	

Address	Classification	Description	Fixed 5/01	5/02	5/03	5/04	5/05
S:99	Dynamic Config	**Global Status Word (SLC 5/04 OS401 only)** Data placed in this memory location is transmitted as the processor's Global Status Word and is sent to all other devices on the DH+ network every time the processor passes the DH+ token.				•	
S:100 to S:163	Static Config	**Global Status File (SLC 5/04 OS401 only)** When a processor passes the DH+ token to the next node, it also sends a 16-bit word called the Global Status Word (S:99 and above). All of the nodes on the network read the Global Status Word sent by each processor and saves the word to memory. Each processor has a table (Global Status File) in memory where global status words from other processors are stored. This table is completely updated every token rotation. (Example: The word from node "x" is placed at S:100 + x.) You can use the Global Status File as a high–speed broadcast message for status passing and synchronization of processors.				•	

C Memory Usage and Instruction Execution Times

This appendix provides:

- instruction words and instruction execution times for the MicroLogix 1000 controllers
- instruction words and instruction execution times for the Fixed, SLC 5/01, SLC 5/02, SLC 5/03, SLC 5/04, and SLC 5/05 processor
- examples on how to estimate the total memory usage of your system

MicroLogix 1000 Controllers

The table below lists the execution times and memory usage for the MicroLogix 1000 controller instructions. Any instruction that takes longer than 15 µs (true or false execution time) to execute performs a poll for user interrupts.

Mnemonic	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
ADD	6.78	33.09	1.50	Add	Math
AND	6.78	34.00	1.50	And	Data Handling
BSL	19.80	53.71 + 5.24 x position value	2.00	Bit Shift Left	Application Specific
BSR	19.80	53.34 + 3.98 x position value	2.00	Bit Shift Right	Application Specific

Mnemonic	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
CLR	4.25	20.80	1.00	Clear	Math
COP	6.60	27.31 + 5.06/word	1.50	File Copy	Data Handling
CTD	27.22	32.19	1.00	Count Down	Basic
CTU	26.67	29.84	1.00	Count Up	Basic
DCD	6.78	27.67	1.50	Decode 4 to 1 of 16	Data Handling
DDV	6.78	157.06	1.00	Double Divide	Math
DIV	6.78	147.87	1.50	Divide	Math
ENC	6.78	54.80	1.50	Encode 1 of 16 to 4	Data Handling
EQU	6.60	21.52	1.50	Equal	Comparison
FFL	33.67	61.13	1.50	FIFO Load	Data Handling
FFU	34.90	73.78 + 4.34 x position value	1.50	FIFO Unload	Data Handling
FLL	6.60	26.86 + 3.62/word	1.50	Fill File	Data Handling
FRD	5.52	56.88	1.00	Convert from BCD	Data Handling
GEQ	6.60	23.60	1.50	Greater Than or Equal	Comparison
GRT	6.60	23.60	1.50	Greater Than	Comparison
HSC	21.00	21.00	1.00	High–Speed Counter	High–Speed Counter
HSD	7.00	8.00	1.25	High–Speed Counter Interrupt Disable	High–Speed Counter
HSE	7.00	10.00	1.25	High–Speed Counter Interrupt. Enable	High–Speed Counter
HSL	7.00	66.00	1.50	High–Speed Counter Load	High–Speed Counter
IIM	6.78	35.72	1.50	Immediate Input with Mask	Program Flow Control
INT	0.99	1.45	0.50	Interrupt Subroutine	Application Specific
IOM	6.78	41.59	1.50	Immediate Output with Mask	Program Flow Control
JMP	6.78	9.04	1.00	Jump to Label	Program Flow Control
JSR	4.25	22.24	1.00	Jump to Subroutine	Program Flow Control

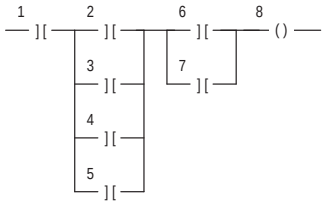
Mnemonic	False Execution Time (μs)	True Execution Time (μs)	Memory Usage (user words)	Name	Instruction Type
LBL	0.99	1.45	0.50	Label	Program Flow Control
LEQ	6.60	23.60	1.50	Less Than or Equal	Comparison
LES	6.60	23.60	1.50	Less Than	Comparison
LIM	7.69	36.93	1.50	Limit Test	Comparison
LFL	33.67	61.13	1.50	LIFO Load	Data Handling
LFU	35.08	64.20	1.50	LIFO Unload	Data Handling
MCR	4.07	3.98	0.50	Master Control Reset	Program Flow Control
MEQ	7.69	28.39	1.50	Masked Comparison for Equal	Comparison
MOV	6.78	25.05	1.50	Move	Data Handling
MSG	26	180①②	34.75	Message	Communication
MUL	6.78	57.96	1.50	Multiply	Math
MVM	6.78	33.28	1.50	Masked Move	Data Handling
NEG	6.78	29.48	1.50	Negate	Data Handling
NEQ	6.60	21.52	1.50	Not Equal	Comparison
NOT	6.78	28.21	1.00	Not	Data Handling
OR	6.78	33.68	1.50	Or	Data Handling
OSR	11.48	13.02	1.00	One–Shot Rising	Basic
OTE	4.43	4.43	0.75	Output Energize	Basic
OTE (high–speed counter)	7.00	12.00	0.75	Update High–Speed Counter Image Accumulator	High–Speed Counter
OTL	3.16	4.97	0.75	Output Latch	Basic
OTU	3.16	4.97	0.75	Output Unlatch	Basic
RAC	6.00	56.00	1.00	High–Speed Counter Reset Accumulator	High–Speed Counter
RES (timer/ counter)	4.25	15.19	1.00	Reset	Basic
RES (high–speed counter)	6.00	51.00	1.00	High–Speed Counter Reset	High–Speed Counter

Mnemonic	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
RET	3.16	31.11	0.50	Return from Subroutine	Program Flow Control
RTO	27.49	38.34	1.00	Retentive Timer	Basic
SBR	0.99	1.45	0.50	Subroutine	Program Flow Control
SCL	6.78	169.18	1.75	Scale Data	Math
SQC	27.40	60.52	2.00	Sequencer Compare	Application Specific
SQL	28.12	53.41	2.00	Sequencer Load	Application Specific
SQO	27.40	60.52	2.00	Sequencer Output	Application Specific
SQR	6.78	71.25	1.25	Square Root	Math
STD	3.16	6.69	0.50	Selectable Timer Interrupt Disable	Application Specific
STE	3.16	10.13	0.50	Selectable Timer Interrupt Enable	Application Specific
STS	6.78	24.59	1.25	Selectable Timer Interrupt Start	Application Specific
SUB	6.78	33.52	1.50	Subtract	Math
SUS	7.87	10.85	1.50	Suspend	Program Flow Control
TND	3.16	7.78	0.50	Temporary End	Program Flow Control
TOD	6.78	49.64	1.00	Convert to BCD	Data Handling
TOF	31.65	39.42	1.00	Timer Off–Delay	Basic
TON	30.38	38.34	1.00	Timer On–Delay	Basic
XIC	1.72	1.54	0.75	Examine If Closed	Basic
XIO	1.72	1.54	0.75	Examine If Open	Basic
XOR	6.92	33.64	1.50	Exclusive Or	Data Handling

① This only includes the amount of time needed to set up the operation requested. It does not include the time it takes to service the actual communication, as this time varies with each network configuration. For example, 144 ms is the actual communication service time for the following configuration 3 nodes on DH-485 (2 MicroLogix and 1 PC for programming), running at 19.2K baud with 2 words per transfer.

② Add 7.3 microseconds per word for MSG instructions that perform writes.

MicroLogix 1000 Execution Time Example



For the rung shown above:

1. If instruction 1 is false, instructions 2, 3, 4, 5, 6, 7 take zero execution time.
   Execution time = 1.54 + 12 = 13.54 microseconds

2. If instruction 1 is true, instruction 2 is true, and 6 is true, then instructions 3, 4, 5, and 7 take zero execution time.
   Execution time = 1.54
   1.54
   1.54
   12.0
   16.62 microseconds

3. If instructions 1, 3, and 6 are true, then instruction 2 takes a “false” execution time, but instructions 4, 5, and 7 take zero execution time.
   Execution time = 1.54
   1.72
   1.54
   1.54
   12.0
   18.34 microseconds

Estimating Memory Usage for Your MicroLogix 1000 Control System

Use the following to calculate memory usage for your MicroLogix 1000 control system.

_____	**1.** Determine the total instruction words used by the instructions in your program and enter the result. Refer to the table on page C–1.
_____	**2.** Multiply the total number of rungs by 0.75 and enter the result. Do not count the END rungs in each file.
177	**3.** To account for controller overhead, use 177.
110	**4.** To account for application data, use 110.
Total Memory Usage: _____	**5.** Total steps 1–4. This is the estimated total memory usage of your application system. Remember, this is an estimate, actual compiled programs may differ by ±12%.
	6. To determine the estimated amount of memory remaining in the controller you have selected, do the following:
1024	Subtract the total memory usage from 1024.
Total Memory Usage (from above): – _____	
Total Memory Remaining: _____	The result of this calculation will be the estimated total memory remaining in your selected controller.

Note *The calculated memory usage may vary from the actual compiled program by ±12%.*

MicroLogix 1000 Memory Usage Example

MicroLogix 1000 Controller

```
  50 XIC and XIO                50 x 0.75 = 37.50
  15 OTE instructions           15 x 0.75 = 11.25
  5 TON instructions             5 x 1.00 =  5.00
  3 GRT instructions             3 x 1.50 =  4.50
  1 SCL instruction              1 x 1.75 =  1.75
  1 TOD instruction              1 x 1.00 =  1.00
  3 MOV instructions             3 x 1.50 =  4.50
  10 CTU instructions           10 x 1.00 = 10.00
  10 RES instructions           10 x 1.00 = 10.00
Instruction Usage                            85.50

  30 rungs                       30 x.750 = 22.50
  controller overhead =                    177.00
  application data                         110.00
```

Estimated Total Memory Usage **395.00**

```
1024 – 395 = 629 instruction words remaining
in the processor
```

Memory Usage Overview for the SLC Processors

SLC 500 controllers have the following user memory capacities:

Type of Processor	User Memory Capacity
Fixed and SLC 5/01	1024 instruction words
SLC 5/02	4096 instruction words
SLC 5/03	12,288 words
SLC 5/04 SLC 5/05	20,480 words①

① When your ladder program is larger than 12K words, you must split your program into two files. A main (file 2) and at least one subroutine file (3–255) is required.

The following definitions apply when figuring your memory usage:

- fixed, SLC 5/01, and SLC 5/02 – 1 instruction word = 4 data words = 8 bytes
- SLC 5/03, SLC 5/04, and SLC 5/05 – 1 instruction word = 1 data word

Fixed and SLC 5/01 Processors

The number of words used by an instruction is indicated in the following table. Since the program is compiled by the programmer, it is only possible to establish *estimates* for the instruction words used by individual instructions. The calculated memory usage will normally be greater than the actual memory usage, due to compiler optimization.

Mnemonic	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
ADD	12	122	1.5	Add	Math
AND	12	87	1.5	And	Data Handling
BSL	12	144 + 24 per word	2.00	Bit Shift Left	Application Specific
BSR	12	144 + 24 per word	2.00	Bit Shift Right	Application Specific
CLR	12	40	1.00	Clear	Math
COP	12	45 + 21 per word	1.50	File Copy	Data Handling
CTD	12	111	1.00	Count Down	Basic
CTU	12	111	1.00	Count Up	Basic
DCD	12	80	1.50	Decode 4 to 1 of 16	Data Handling
DDV	12	650	1.00	Double Divide	Math
DIV	12	400	1.50	Divide	Math
EQU	12	60	1.50	Equal	Comparison
FLL	12	37 + 14 per word	1.50	Fill File	Data Handling
FRD	12	223	1.00	Convert from BCD	Data Handling
GEQ	12	60	1.50	Greater Than or Equal	Comparison
GRT	12	60	1.50	Greater Than	Comparison
HSC	12	60	1.00	High–Speed Counter	High–Speed Counter
IIM	12	372	1.50	Immediate Input with Mask	Program Flow Control
IOM	12	475	1.50	Immediate Output with Mask	Program Flow Control
JMP	12	38	1.00	Jump to Label	Program Flow Control
JSR	12	46	1.00	Jump to Subroutine	Program Flow Control
LBL	2	2	0.50	Label	Program Flow Control
LEQ	12	60	1.50	Less Than or Equal	Comparison

Mnemonic	False Execution Time (μs)	True Execution Time (μs)	Memory Usage (user words)	Name	Instruction Type
LES	12	60	1.50	Less Than	Comparison
MCR	10	10	0.50	Master Control Reset	Program Flow Control
MEQ	12	75	1.50	Masked Compare for Equal	Comparison
MOV	12	20	1.50	Move	Data Handling
MUL	12	230	1.50	Multiply	Math
MVM	12	115	1.50	Masked Move	Data Handling
NEG	12	110	1.50	Negate	Data Handling
NEQ	12	60	1.50	Not Equal	Comparison
NOT	12	66	1.00	Not	Data Handling
OR	12	87	1.50	Or	Data Handling
OSR	12	34	1.00	One–Shot Rising	Basic
OTE	18	18	0.75	Output Energize	Basic
OTL	19	19	0.75	Output Latch	Basic
OTU	19	19	0.75	Output Unlatch	Basic
RES	12	40	1.00	Reset	Basic
RET	12	34	0.50	Return from Subroutine	Program Flow Control
RTO	12	140	1.00	Retentive Timer	Basic
SBR	2	2	0.50	Subroutine	Program Flow Control
SQC	12	225	2.00	Sequencer Compare	Application Specific
SQO	12	225	2.00	Sequencer Output	Application Specific
SUB	12	125	1.50	Subtract	Math
SUS	12	12	1.50	Suspend	Program Flow Control
TND	12	32	0.50	Temporary End	Program Flow Control
TOD	12	200	1.00	Convert to BCD	Data Handling
TOF	12	140	1.00	Timer Off–Delay	Basic
TON	12	135	1.00	Timer On–Delay	Basic
XIC	4	4	1.00	Examine If Closed	Basic
XIO	4	4	1.00	Examine If Open	Basic

Mnemonic	False Execution Time (μs)	True Execution Time (μs)	Memory Usage (user words)	Name	Instruction Type
XOR	12	87	1.50	Exclusive Or	Data Handling

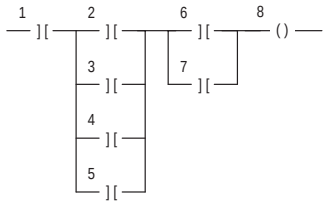
① These instructions take zero execution time if they are preceded by conditions that guarantee the state of the rung. Rung logic is solved left to right. Branches are solved top to bottom.

Fixed or SLC 5/01 Execution Time Example

For the rung example below:

1) If instruction 1 is false, instructions 2, 3, 4, 5, 6, 7 take zero execution time. Execution time = 4 + 18 = 22 microseconds.

2) If instruction 1 is true, 2 is true, and 6 is true, then instructions 3, 4, 5, 7 take zero execution time. Execution time = 4 + 4 + 4 + 4 + 18 = 30 microseconds.



Estimating Total Memory Usage of Your System Using a Fixed or SLC 5/01 Processor

_____ **1.** Calculate the total instruction words used by the instructions in your program and enter the result. Refer to the table on page C–8.

_____ **2.** Multiply the total number of rungs by .375 and enter the result.

_____ **3.** Multiply the total number of data words (excluding the status file and I/O data words) by .25 and enter the result.

_____ **4.** Add 1 word for each data table file and enter the result.

_____ **5.** Multiply the highest numbered program file used by 2 and enter the result.

_____ **6.** Multiply the total number of I/O data words by .75 and enter the result.

_____ **7.** Multiply the total number of I/O slots, used or unused, by .75 and enter the result.

_____ **8.** To account for processor overhead, enter 65 if you are using a fixed controller; enter 67 if you are using a 1747–L511 or 1747–L514.

Total: _____ **9.** Total steps 1 through 8. This is the estimated total memory usage of your application system. Remember, this is an estimate, actual compiled programs may differ by ±12%.

10. If you wish to determine the estimated amount of memory remaining in the processor you have selected, do the following:

If you are using a fixed controller or 1747–L511, subtract the total from 1024. If you are using a 1747–L514, subtract the total from 4096.

The result of this calculation will be the estimated total memory remaining in your selected processor.

Note *The calculated memory usage may vary from the actual compiled program by ±12%.*

Fixed Controller Memory Usage Example

L20B Fixed I/O Controller

42 XIC and XIO	42 x 1.00 =	42.00
10 OTE instructions	10 x 0.75 =	7.50
10 TON instructions	10 x 1.00 =	10.00
1 CTU instruction	1 x 1.00 =	1.00
1 RES instruction	1 x 1.00 =	1.00
Instruction Usage		61.50
21 rungs	21 x.375 =	7.87
37 data words	37 x.250 =	9.25
User Program Total		**78.62**
2 I/O data words	2 x 0.75 =	1.50
1 slot	1 x 0.75 =	0.75
Overhead		65.00
I/O Configuration Total		**67.25**
Estimated total memory usage:		145.87
		(round to 146)

1024 – 146 = 878 instruction words remaining in the processor

SLC 5/01 Processor Memory Usage Example

1747-L514 processor, 30-slot configuration, (15) 1746-IA16, (10) 1746-OA8, (1) 1747-DCM full configuration, (1) 1746-NI4, (1) 1746-NIO4I

50 XIC and XIO	50 x 1.00 = 50.00
15 OTE instructions	15 x 0.75 = 11.25
5 TON instructions	5 x 1.00 = 5.00
3 GRT instructions	3 x 1.50 = 4.50
1 SCL instruction	1 x 1.75 = 1.75
1 TOD instruction	1 x 1.00 = 1.00
3 MOV instructions	3 x 1.50 = 4.50
10 CTU instructions	10 x 1.00 = 10.00
10 RES instructions	10 x 1.00 = 10.00
Instruction Usage	98.00
30 rungs	30 x 0.375 = 11.25
100 data words	100 x 0.25 = 25.00
10 is highest data table file number	10 x 1 = 10.00
4 is highest program file number	4 x 2 = 8.00
User Program Total	**163.50**
49 I/O data words	49 x 0.75 = 36.75
30 slot	30 x 0.75 = 22.50
Overhead	67.00
I/O Configuration Total	**126.25**
Estimated total memory usage:	289.75
	(round to 290)

4096 – 290 = 3806 instruction words remaining in processor

SLC 5/02 Processor

The number of instruction words used by an instruction is indicated in the following table. Since the program is compiled by the programmer, it is only possible to establish *estimates* for the instruction words used by individual instructions. The calculated memory usage will normally be greater than the actual memory usage, due to compiler optimization.

Mnemonic	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
ADD	7	76	1.5	Add	Math
AND	7	55	1.5	And	Data Handling
BSL	36	89 +14 per word	2.00	Bit Shift Left	Application Specific
BSR	36	83 +14 per word	2.00	Bit Shift Right	Application Specific
CLR	7	26	1.00	Clear	Math
COP	7	29 + 13 per word	1.50	File Copy	Data Handling
CTD	7	69	1.00	Count Down	Basic
CTU	7	69	1.00	Count Up	Basic
DCD	7	50	1.50	Decode 4 to 1 of 16	Data Handling
DDV	7	392	1.00	Double Divide	Math
DIV	7	242	1.50	Divide	Math
EQU①	38	38	1.50	Equal	Comparison
FFL	51	150	1.50	FIFO Load	Data Handling
FFU	51	150	1.50	FIFO Unload	Data Handling
FLL	7	25 + 8 per word	1.50	Fill File	Data Handling
FRD	7	136	1.00	Convert from BCD	Data Handling
GEQ①	38	38	1.50	Greater Than or Equal	Comparison
GRT①	38	38	1.50	Greater Than	Comparison
IID	7	39	1.25	I/O Interrupt Disable	Interrupt
IIE	7	42	1.25	I/O Interrupt Enable	Interrupt
IIM	1	340	1.50	Immediate Input with Mask	Program Flow Control
INT	0	0	0.50	Interrupt Subroutine	Application Specific
IOM	7	465	1.50	Immediate Output with Mask	Program Flow Control

Mnemonic	False Execution Time (μs)	True Execution Time (μs)	Memory Usage (user words)	Name	Instruction Type
JMP	7	23	1.00	Jump to Label	Program Flow Control
JSR	7	28	1.00	Jump to Subroutine	Program Flow Control
LBL	1	4	0.50	Label	Program Flow Control
LEQ①	38	38	1.50	Less Than or Equal	Comparison
LES①	38	38	1.50	Less Than	Comparison
LIM	7	45	1.50	Limit Test	Comparison
LFL	51	150	1.50	LIFO Load	Data Handling
LFU	51	180	1.50	LIFO Unload	Data Handling
MCR	6	6	0.50	Master Control Reset	Program Flow Control
MEQ①	7	47	1.50	Masked Comparison for Equal	Comparison
MOV	7	14	1.50	Move	Data Handling
MSG	48	180②	34.75	Message	Communication
MUL	7	140	1.50	Multiply	Math
MVM	7	71	1.50	Masked Move	Data Handling
NEG	7	68	1.50	Negate	Data Handling
NEQ①	38	38	1.50	Not Equal	Comparison
NOT	7	42	1.00	Not	Data Handling
OR	7	55	1.50	Or	Data Handling
OSR	11	20	1.00	One–Shot Rising	Basic
OTE	11	11	0.75	Output Energize	Basic
OTL	11	11	0.75	Output Latch	Basic
OTU	11	11	0.75	Output Unlatch	Basic
PID	90	3600	23.25	Proportional Derivative	PID
REF	4	240	0.50	Refresh	Interrupt
RES	7	26	1.00	Reset	Basic
RET	7	20	0.50	Return from Subroutine	Program Flow Control
RPI	7	240	1.25	Reset Pending Interrupt	Interrupt
RTO	30	86	1.00	Retentive Timer	Basic

Mnemonic	False Execution Time (μs)	True Execution Time (μs)	Memory Usage (user words)	Name	Instruction Type
SBR	1	4	0.50	Subroutine	Program Flow Control
SCL	7	480	1.75	Scale Data	Math
SQC	36	137	2.00	Sequencer Compare	Application Specific
SQL	36	135	2.00	Sequencer Load	Application Specific
SQO	36	137	2.00	Sequencer Output	Application Specific
SQR	7	162	1.25	Square Root	Math
STD	4	9	0.50	Selectable Timer Interrupt Disable	Application Specific
STE	4	9	0.50	Selectable Timer Interrupt Enable	Application Specific
STS	7	72	1.25	Selectable Timer Interrupt Start	Application Specific
SUB	7	77	1.50	Subtract	Math
SUS	7	7	1.50	Suspend	Program Flow Control
SVC	4	240		Service Comms	Communication
TND	7	22	0.50	Temporary End	Program Flow Control
TOD	7	122	1.00	Convert to BCD	Data Handling
TOF	36	86	1.00	Timer Off–Delay	Basic
TON	36	83	1.00	Timer On–Delay	Basic
XIC①	2.40	2.4	1.00	Examine If Closed	Basic
XIO①	2.40	2.4	1.00	Examine If Open	Basic
XOR	7	55	1.50	Exclusive Or	Data Handling

① These instructions take zero execution time if they are preceded by conditions that guarantee the state of the rung. Rung logic is solved left to right. Branches are solved top to bottom.

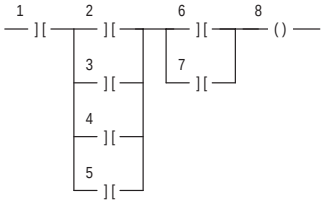
② This only includes the amount of time needed to “set up” the operation requested. It does not include the time it takes to service the actual communications.

SLC 5/02 Execution Time Example

For the rung example below:

1) If instruction 1 is false, instructions 2, 3, 4, 5, 6, 7 take zero execution time.
   Execution time = 2.4 + 11 = 13.4 microseconds.

2) If instruction 1 is true, 2 is true, and 6 is true, then instructions 3, 4, 5, 7 take zero execution time. Execution time = 2.4 + 2.4 + 2.4 + 11 = 18.2 microseconds.



Estimating Total Memory Usage of Your System Using a SLC 5/02 Processor

_____ **1.** Calculate the total instruction words used by the instructions in your program and enter the result. Refer to the table on page C–14.

_____ **2.** Multiply the total number of rungs by .375 and enter the result.

_____ **3.** If you are using a 1747–L524 and have enabled the Single Step Test mode, multiply the total number of rungs by .375 and enter the result.

_____ **4.** Multiply the total number of data words (excluding the status file and I/O data words) by .25 and enter the result.

_____ **5.** Add 1 word for each data table file used and enter the result.

_____ **6.** Multiply the highest numbered program file used by 2 and enter the result.

_____ **7.** Multiply the total number of I/O data words by .75 and enter the result.

_____ **8.** Multiply the total number of I/O slots, used or unused, by .75 and enter the result.

_____ **9.** To account for processor overhead, enter 204.

Total: _____ **10.** Total steps 1 through 9. This is the estimated total memory usage of your application system. Remember, this is an estimate, actual compiled programs may differ by ±12%.

11. If you wish to determine the estimated amount of memory remaining in the processor you have selected, do the following:

If you are using a 1747–L524, subtract the total from 4096.

The result of this calculation will be the estimated total memory remaining in your selected processor.

Note *The calculated memory usage may vary from the actual compiled program by ±12%.*

SLC 5/02 Memory Usage Example

1747-L524 series C processor, 30-slot configuration, (15) 1746-IA16,(10) 1746-OA8, (1) 1747-DCM full configuration, (1) 1746-NI4, (1) 1746-NIO4I

50 XIC and XIO	50 x 1.00 = 50.00
15 OTE instructions	15 x 0.75 = 11.25
5 TON instructions	5 x 1.00 = 5.00
3 GRT instructions	3 x 1.50 = 4.50
1 SCL instruction	1 x 1.75 = 1.75
1 TOD instruction	1 x 1.00 = 1.00
3 MOV instructions	3 x 1.50 = 4.50
10 CTU instructions	10 x 1.00 = 10.00
10 RES instructions	10 x 1.00 = 10.00
Instruction Usage	98.00
30 rungs	30 x 0.375 = 11.25
100 data words	100 x 0.25 = 25.00
10 is highest data table file number	10 x 1 = 10.00
4 is highest program file number	4 x 2 = 8.00
User Program Total	**163.50**
49 I/O data words	49 x 0.75 = 36.75
30 slot	30 x 0.75 = 22.50
Overhead	204.00
I/O Configuration Total	**263.25**
Estimated total memory usage:	426.75
	(round to 427)

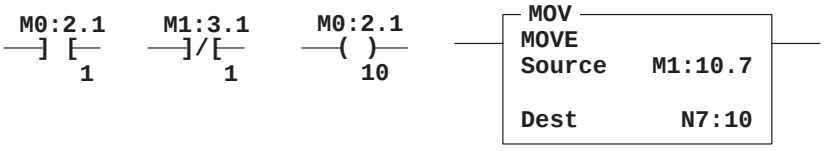
4096 – 427 = 3669 instruction words remaining in processor

SLC 5/02 – Instructions Having Indexed Addresses

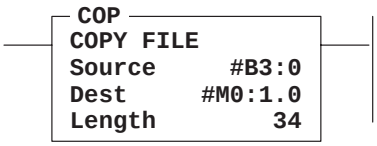
For each operand having an indexed address, add 30 microseconds to the execution time for a true instruction. For example, if a MOV instruction has an indexed address for both the source and destination, the execution time when the instruction is true is 14 + 30 + 30 = 74 microseconds.

SLC 5/02 – Instructions Having M0 and M1 Data File Addresses

For each bit or word instruction, add 1157 microseconds to the execution time. For each multiple-word instruction, add 950 microseconds plus 400 microseconds per word.



Example



For the multi–word instruction above, add 950 microseconds plus 400 microseconds per word. In this example, 34 words are copied from #B:3.0 to M0:1.0. Add 950 + (400 x 34) = 14550 microseconds to the execution time listed on page C–14. This comes to 471 plus 14550 = 15021 microseconds total, or 15.0 milliseconds.

User Word Comparison Between SLC 5/03 (and higher) Processors and the SLC 5/02 Processor

The SLC 5/03 (and higher) processors and the SLC 5/02 processor accumulate user words differently during the creation of a user program. The SLC 5/02 processor is generally more efficient in its word usage than the SLC 5/03 (and higher) processors. However, the SLC 5/02 processor word usage is difficult to estimate since it is tied to the architecture of the microprocessor.

The SLC 5/03 (and higher) processors accumulates words in a way that is easier to understand and estimate than the SLC 5/02 processor. The SLC 5/03 (and higher) processors accumulate words similar to a PLC-5. The SLC 5/03 processor offers 12,288 words. The SLC 5/04 and SLC 5/05 processors offers 20,480 words.

The SLC 5/02 processor offers 4096 user words. It is important to realize that this does not mean that a SLC 5/03 processor can hold a user program that is three times larger than an equivalent SLC 5/02 program. Use the information below to determine the SLC 5/03 user program size based on existing SLC 5/02 programs.

Instruction Words

Some instructions use the same amount of memory, while other instructions do not use the same amount of memory. For example, a CTU instruction always uses 1 word. However, an ADD instruction in a SLC 5/02 processor uses 1.5 words; in a SLC 5/03 (or higher) processor an ADD instruction uses 3 words. Also note additional differences below:

Condition	SLC 5/02 Words	SLC 5/03 Words	SLC 5/04 and SLC 5/05 Words
Each rung	0.375	1	1
Each additional Program File	1	5	5
Each additional Data File	1	5	5
Each I/O Slot	0.75	3	3
Overhead	216	236	250

Exact program content determines the program size difference. An SLC 5/03 program consumes 20% to 150% more instruction words than its SLC 5/02 equivalent.

Data Words

Files 0 and 1

In the SLC 5/02 processor, each I/O data word consumes 0.75 words of memory. In the SLC 5/03 processor, each I/O data word consumes 3 words of data.

File 2

The status file word usage is contained in the overhead values for both the SLC 5/02 and SLC 5/03 processors.

File 3 to 255

In the SLC 5/02 processor, 4 data words consume the same amount of memory as 1 instruction word. This is why the SLC 5/02 processor is said to offer 4K of Instruction words or 16K of Data words. This dynamic amount of Data word storage is due to the architecture of the SLC 5/02's microprocessor.

SLC 5/03 Processor

The following table shows memory usage and instruction execution times for the SLC 5/03 processor. Instructions that support floating point are included within this table. When using a SLC 5/03 processor, it is important to remember that 1 instruction word equals 1 data word.

Mnemonic FP = floating point	Applies to SLC 5/03			False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
	OS300	OS301	OS302					
ABL		•	•	39.70	129.90	2.00	Test Buffer for Line	ASCII
ABS	•	•	•	0.75	9.95	2.00	Absolute	Math
ABS FP			•	0.75	5.20	2.00	Absolute	Math
ACB		•	•	39.70	140.7	2.00	Number of Characters in Buffer	ASCII
ACI		•	•	0.22	86.62	2.00	String to Integer	ASCII
ACL		•	•	0.22	367.50	2.00	ASCII Clear Recieve and/or Send Buffer	ASCII
ACN		•	•	0.22	69.4 + (2.1/char)	3.00	String Concatenate	ASCII
ACS① FP			•	0.75	510.85	2.00	Arc Cosine	Math
ADD	•	•	•	0.75	1.70	3.00, 4.00	Add	Math
ADD FP			•	0.22	38.44	4.00	Add	Math
AEX		•	•	0.22	56.2 + (4.7/char)	4.00	String Extract	ASCII
AHL		•	•	39.70	138.70	4.00	ASCII Handshake Lines	ASCII
AIC		•	•	0.22	103.40	2.00	Integer to String	ASCII
AND	•	•	•	0.75	1.70	3.00	And	Data Handling
ARD		•	•	39.70	181.8	3.00	ASCII Read Characters	ASCII
ARL		•	•	39.70	190.00	3.00	ASCII Read Line	ASCII
ASC		•	•	0.22	53.4 + (1.8/char)	4.00	String Search	ASCII
ASN① FP			•	0.75	483.05	2.00	Arc Sine	Math
ASR		•	•	0.22	49.69	3.00	ASCII String Compare	ASCII
ATN① FP			•	0.75	387.05	2.00	Arc Tangent	Math

Mnemonic FP = floating point	Applies to SLC 5/03 OS300	OS301	OS302	False Execution Time (μs)	True Execution Time (μs)	Memory Usage (user words)	Name	Instruction Type
AWA		•	•	39.7	365.50	3.00	ASCII Write with Append	ASCII
AWT		•	•	39.7	263.80	3.00	ASCII Write	ASCII
BSL	•	•	•	15.00	50 + (2.3/word)	3.00	Bit Shift Left	Application Specific
BSR	•	•	•	15.00	50 + (2.3/word)	3.00	Bit Shift Right	Application Specific
CLR	•	•	•	0.75	1.70	3.00, 1.00	Clear	Math
CLR FP			•	0.22	6.62	1.00	Clear	Math
COP	•	•	•	0.75	30 + (2.20/word)	3.00	File Copy	Data Handling
COS① FP			•	0.75	310.90	2.00	Cosine	Math
CPT① FP			•	0.75	8.8②	③	Compute	Math
CTD	•	•	•	1.40	1.40	1.00	Count Down	Basic
CTU	•	•	•	1.40	1.40	1.00	Count Up	Basic
DCD	•	•	•	0.50	10.00	2.00	Decode 4 to 1 of 16	Data Handling
DDV		•	•	0.50	33.00	2.00	Double Divide	Math
DEG① FP			•		32.80	2.00	Degree	Data Handling
DIV	•	•	•	0.75	23.00	3.00, 4.00	Divide	Math
DIV FP			•	0.22	57.56	4.00	Divide	Math
EQU④	•	•	•	1.25	1.25	3.00	Equal	Comparison
EQU FP		•	•	0.75	12.94	3.00	Equal	Comparison
FFL	•	•	•	27.00	58.00	3.00	FIFO Load	Data Handling
FFU	•	•	•	27.00	79 + (2.20/word)	4.00	FIFO Unload	Data Handling
FLL	•	•	•	0.75	28 + (2.00/word)	3.00	Fill File	Data Handling
FRD	•	•	•	0.50	31.00	2.00	Convert from BCD	Data Handling
GEQ④	•	•	•	1.25	1.25	3.00	Greater Than or Equal	Comparison
GEQ FP		•	•	0.75	14.81	3.00	Greater Than or Equal	Comparison

Mnemonic FP = floating point	Applies to SLC 5/03 OS300	OS301	OS302	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
GRT④	•	•	•	1.25	1.25	3.00	Greater Than	Comparison
GRT FP		•	•	0.75	14.82	3.00	Greater Than	Comparison
IID	•	•	•	0.50	6.00	2.00	I/O Interrupt Disable	Interrupt
IIE	•	•	•	0.50	16.00	2.00	I/O Interrupt Enable	Interrupt
IIM⑤	•	•	•	0.50	51.85	6.00	Immediate Input with Mask	Program Flow Control
INT	•	•	•	0.25	0.25	1.00	Interrupt Subroutine	Application Specific
IOM⑤	•	•	•	0.50	70.90	6.00	Immediate Output with Mask	Program Flow Control
JMP	•	•	•	0.25	44.45	1.00	Jump to Label	Program Flow Control
JSR	•	•	•	0.25	131.00	1.00	Jump to Subroutine	Program Flow Control
LBL	•	•	•	0.25	0.25	2.00	Label	Program Flow Control
LEQ④	•	•	•	1.25	1.25	3.00	Less Than or Equal	Comparison
LEQ FP		•	•	0.75	13.19	3.00	Less Than or Equal	Comparison
LES④	•	•	•	1.25	1.25	3.00	Less Than	Comparison
LES FP		•	•	0.75	13.19	3.00	Less Than	Comparison
LFL	•	•	•	27.00	58.00	3.00	LIFO Load	Data Handling
LFU	•	•	•	27.00	66.00	3.00	LIFO Unload	Data Handling
LIM③	•	•	•	1.95	1.95	1.00	Limit Test	Comparison
LIM FP		•	•	0.75	22.81	1.00	Limit Test	Comparison
LN① FP			•	0.75	392.00	2.00	Natural Log	Math
LOG① FP			•	0.75	390.80	2.00	Log to the Base 10	Math
MCR	•	•	•	8.00	4.00	1.00	Master Control Reset	Program Flow Control
MEQ④	•	•	•	38.00	38.00	4.00	Masked Comparison for Equal	Comparison

Mnemonic FP = floating point	Applies to SLC 5/03 OS300	Applies to SLC 5/03 OS301	Applies to SLC 5/03 OS302	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
MOV	•	•	•	0.50	1.25	2.00	Move	Data Handling
MOV FP		•	•	0.22	12.19	2.00	Move	Data Handling
MSG⑥	•	•	•	60.00	203.00	20.00	Message	Communication
MUL	•	•	•	0.75	20.00	3.00	Multiply	Math
MUL FP			•	0.22	39.05	3.00	Multiply	Math
MVM	•	•	•	0.75	19.00	3.00, 4.00	Masked Move	Data Handling
NEG	•	•	•	0.75	1.70	3.00	Negate	Data Handling
NEG FP			•	0.22	12.38	3.00	Negate	Data Handling
NEQ④	•	•	•	1.25	1.25	3.00	Not Equal	Comparison
NEQ FP		•	•	0.75	13.25	3.00	Not Equal	Comparison
NOT	•	•	•	0.75	1.70	3.00	Not	Data Handling
OR	•	•	•	0.75	1.70	3.00	Or	Data Handling
OSR	•	•	•	12.00	10.80	2.00	One–Shot Rising	Basic
OTE	•	•	•	0.63	0.63	1.00	Output Energize	Basic
OTL	•	•	•	0.63	0.63	1.00	Output Latch	Basic
OTU	•	•	•	0.63	0.63	1.00	Output Unlatch	Basic
PID	•	•	•	20.00	272.00	26.00	Proportional Integral Derivative	PID
RAD① FP			•	0.75	31.80	2.00	Radian	Data Handling
REF	•	•	•	0.25	240.00⑦	1.00	I/O Refresh	Program Flow Control
RES	•	•	•	1.40	1.40	1.00	Reset	Basic
RET	•	•	•	0.25	23.00	1.00	Return from Subroutine	Program Flow Control
RPI	•	•	•	0.50	78 + (60/slot)	2.00	Reset Pending Interrupt	Interrupt
RTO	•	•	•	1.40	1.40	1.00	Retentive Timer	Basic
SBR	•	•	•	0.25	0.25	1.00	Subroutine	Program Flow Control
SCL① FP			•	1.00	32.00	4.00	Scale Data	Math

Mnemonic FP = floating point	Applies to SLC 5/03 OS300	Applies to SLC 5/03 OS301	Applies to SLC 5/03 OS302	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
SCP			•	0.75	33.10	6.00	Scale with Parameters	Math
SCP FP			•	0.75	196.10	6.00	Scale with Parameters	Math
SIN① FP			•	0.75	311.95	2.00	Sine	Math
SQC	•	•	•	13.00	60.00	5.00	Sequencer Compare	Application Specific
SQL	•	•	•	15.00	56.00	4.00	Sequencer Load	Application Specific
SQO	•	•	•	15.00	70.00	5.00	Sequencer Output	Application Specific
SQR	•	•	•	0.50	32.00	2.00, 3.00	Square Root	Math
SQR FP			•	0.22	70.00	3.00	Square Root	Math
STD	•	•	•	0.25	4.00	1.00	Selectable Timer Interrupt Disable	Application Specific
STE	•	•	•	0.25	5.00	1.00	Selectable Timer Interrupt Enable	Application Specific
STS	•	•	•	0.75	58.00	3.00	Selectable Timer Interrupt Start	Application Specific
SUB	•	•	•	0.75	1.70	3.00	Subtract	Math
SUB FP			•	0.22	38.19	4.00	Subtract	Math
SUS	•	•	•	0.50	12.00	2.00	Suspend	Program Flow Control
SVC	•	•	•	0.25	240.00⑧	1.00	Service Communication	Communication
SWP			•	0.75	24 + 13.09/word	2.00	SWAP	Math
TAN① FP			•	0.75	406.35	2.00	Tangent	Math
TND	•	•	•	0.25	12.00	1.00	Temporary End	Program Flow Control
TOD	•	•	•	0.50	38.00	2.00	Convert to BCD	Data Handling
TOF	•	•	•	1.40	1.40	1.00	Timer Off–Delay	Basic
TON	•	•	•	1.40	1.40	1.00	Timer On–Delay	Basic
XIC④	•	•	•	0.44	0.44	1.00	Examine If Closed	Basic

Mnemonic FP = floating point	Applies to SLC 5/03 OS300	OS301	OS302	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
XIO④	•	•	•	0.44	0.44	1.00	Examine If Open	Basic
XOR	•	•	•	0.75	1.70	3.00	Exclusive Or	Data Handling
XPY① FP			•	0.75	699.30	3.00	X to the Power of Y	Math

① The execution times assume floating point data. If signed integer data is used, add 14 microseconds per instruction execution time.

② To get the total execution time for a CPT instruction, take the CPT execution time plus each additional math instruction execution time, plus the number of math instructions times 3.01. For example, if a CPT instruction calls one ADD and one SUB instruction the calculation is: 8.8 + 1.70 + 1.70 + 2(3.01) = 18.22

③ To calculate the memory usage, do the following: Take 2 plus the number of instruction words for each operation performed plus the number of operations performed in the compute. For example, 2 + ADD + SUB + 2 =10.

④ These instructions take zero execution time if they are preceded by conditions that guarantee the state of the rung. Rung logic is solved left to right. Branches are solved from top to bottom.

⑤ Times listed are for discrete I/O modules. When using 32-point I/O modules, add the following microseconds to the IIM and IOM instructions:

- 15 µs IIM when true
- 30 µs IOM when true

When using the following modules and the IIM instruction in your program, add the following microseconds

- Analog or Thermocouple module inputs, add 450 to 550 µs
- BASIC Module inputs, add 500 to 550 µs
- Other specialty inputs, add 425 to 957 µs

When using the following modules and the IOM instruction in your program, add the following microseconds

- Analog or Thermocouple module inputs, add 390 to 416 µs
- BASIC Module inputs, add 440 to 466 µs
- Other specialty inputs, add 590 to 989 µs

⑥ This only includes the amount of time needed to “set up” the operations requested. It does not include the time it takes to service the actual communications.

⑦ This instruction performs a complete end of scan. This includes an input/output scan, service communication, and housekeeping. See Worksheet E in appendix D for calculating the actual execution time.

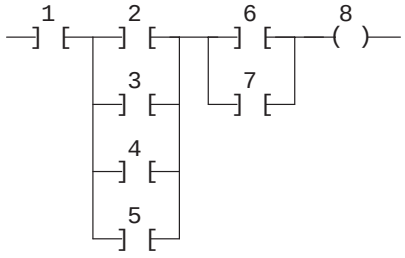
⑧ 1 channel = 150 µs no commands pending
2 channel = 170 µs no commands pending
Add 1 ms per command serviced.

SLC 5/03 Execution Time Example

For the rung example below:

1) If instruction 1 is false, instructions 2, 3, 4, 5, 6, 7 take zero execution time. Execution time = .44 + .63 = 1.07 microseconds.

2) If instruction 1 is true, 2 is true, and 6 is true, then instructions 3, 4, 5, 7 take zero execution time. Execution time = .44 + .44 + .44 + .63 = 1.95 microseconds.



SLC 5/03 Processor Floating Point Operations

Floating point times apply to SLC 5/03 OS301 and higher processors. Refer to the table on page C–23 for the instruction execution times when using floating point operations.

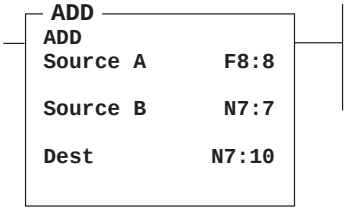
The floating point math times in the following table are applicable when S:34/2 is not set. When S:34/2 is set, the math flags are updated after the instruction is executed and 4.8 μs are added to the execution time. S:34/2 may be changed dynamically (that is, while the program is running, by the program). For each integer parameter, add 10 μs to allow for the integer to floating point conversion.

If you need to perform floating point operations using a mixture of floating point and integer parameters (values and source/destination addresses), calculate the amount of instruction words according to these guidelines:

1. Start with the amount of words for the floating point instruction.
2. Add 2 words for the first integer parameter (value or source/destination address).
3. Add 1 word for every subsequent integer parameter.

These additional words are required for integer-to-floating-point conversions.

For example, this instruction requires 7 instruction words:



- 4 words for the floating point ADD instruction (from the table on page C–30)
- + 2 words for the integer address, N7:7
- + 1 word for the second integer address, N7:10

Estimating Total Memory Usage of Your System Using an SLC 5/03 Processor

_____	**1.**	Add the total number of data file words used (excluding the status file and I/O data words) and enter the result.
_____	**2.**	Multiply the total number of I/O data words by 3 and enter the result.
_____	**3.**	Multiply the total number of I/O slots, used or unused, by 3 and enter the result.
_____	**4.**	To account for processor overhead, enter 236 and enter the result.
_____	**5.**	Multiply the highest numbered data table file used by 5 and enter the result.
_____	**6.**	Multiply the highest numbered program file used by 5 and enter the result.
Subtotal: _____	**7.**	Add steps 1 through 6. Enter this as the subtotal (additional 4K word usage).
4096 – _____ (step 7)	**8.**	Subtract the value in step 7 from 4096; if the result is positive, enter 12,288 in step 14. If the result is negative, subtract the absolute value from 12,288 and enter the result in step 14. (This decreases the value.)
_____		
_____	**9.**	Calculate the total number of words used by the instructions in your program and enter the result. Refer to the table on page C–14.
_____	**10.**	Add the total number of rungs (1 word per rung) and enter the result.
_____	**11.**	Add 1 word for each indexed address reference and enter the result.
_____	**12.**	Add 2 words per rung for each rung that contains an indexed address reference and enter the result.
Subtotal: _____	**13.**	Add steps 9 through 12 and enter the result.
_____	**14.**	Enter the result from step 8. This is the available memory.
_____	**15.**	Enter the result from step 13. This is the total number of words used.
Total: _____	**16.**	Subtract step 15 from step 14. This number is the amount of memory available to your system.

SLC 5/03 Memory Usage Example

1747-L532 processor, 30-slot configuration, (15) 1746-IA16, (10) 1746-OA8, (1) 1747-DCM full configuration, (1) 1746-NI4, (1) 1746-NIO4I

100 data words	100 x 1.00 =	100.00
49 I/O data words	49 x 3.00 =	147.00
30 slot	30 x 3.00 =	90.00
Overhead		236.00
10 is highest data table file number	10 X 5 =	50.00
4 is highest program file number	4 X 5 =	20.00
Subtotal		**643.00**

Account for additional 4K data space

4096 – 643 = 3453 (result is positive; therefore **12,288** words are available)

50 XIC and XIO	50 x 1.00 =	50.00
15 OTE instructions	15 x 1.00 =	15.00
5 TON instructions	5 x 1.00 =	5.00
3 GRT instructions	3 x 3.00 =	9.00
1 SCL instruction	1 x 4.00 =	4.00
1 TOD instruction	1 x 2.00 =	2.00
3 MOV instructions	3 x 2.00 =	6.00
10 CTU instructions	10 x 1.00 =	10.00
10 RES instructions	10 x 1.00 =	10.00
Instruction Usage		**111.00**
30 rungs	30 x 1.00 =	**30.00**
0 indexed address	=	0.00
0 indexed address reference	=	0.00
Subtotal		**141.00**

Available memory	12,288.00
Words used	– 141.00
Estimated total memory available:	**12,147.00**

SLC 5/03 – Instructions Having Indexed Addresses

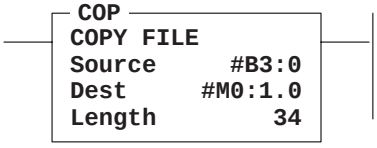
For each operand having an indexed address, add 25 µs microseconds to the execution time for a true instruction. For example, if a MOV instruction has an indexed address for both the source and destination, the execution time when the instruction is true is 1.25 + 25 + 25 = 51.25 microseconds.

SLC 5/03 – Instructions Having M0 and M1 Data File Addresses

Instructions having M0 and M1 data file addresses vary in their execution times. The following execution times represent the expected maximum values.

Instruction Type	Execution Time (µs)
XIC or XIO	782
OTU, OTE, or OTL	925
COP to M file	772 + 23 per word
COP from M file	760 + 22 per word
FLL	753 + 30 per word
MVM to M file	894
any Source or Destination M file address	730

Example



For the multi–word instruction above, add 772 microseconds plus 23 microseconds per word. In this example, 34 words are copied from #B3:0 to M0:1.0. Add 772 + (23 x 34) = 1554 microseconds to the execution time listed on page C–24. This comes to 104.8 (calculated from page C–24 table) plus 1554 = 1658.8 microseconds total, or 1.6 milliseconds.

SLC 5/04 and SLC 5/05 Processors

The following table shows memory usage and instruction execution times for the SLC 5/04 and SLC 5/05 processors. Instructions that support floating point are included within this table. When using SLC 5/04 or SLC 5/05 processors, it is important to remember that 1 instruction word equals 1 data word.

Mnemonic FP = floating point	Applies to SLC 5/04 SLC 5/05 OS400	OS401	OS500	False Execution Time (μs)	True Execution Time (μs)	Memory Usage (user words)	Name	Instruction Type
ABL	•	•	•	35.00	156.00	2.00	Test Buffer for Line	ASCII
ABS	•	•	•	0.56	8.60	2.00	Absolute	Math
ABS FP		•	•	0.75	4.35	2.00	Absolute	Math
ACB	•	•	•	35.00	131.00	2.00	Number of Characters in Buffer	ASCII
ACI	•	•	•	0.18	56.00	2.00	String to Integer	ASCII
ACL	•	•	•	0.18	332.80	2.00	ASCII Clear Recieve and/or Send Buffer	ASCII
ACN	•	•	•	0.18	56 + (2.5/char)	3.00	String Concatenate	ASCII
ACS① FP		•	•	0.56	51.90	2.00	Arc Cosine	Math
ADD	•	•	•	0.56	1.50	3.00, 4.00	Add	Math
ADD FP	•	•	•	0.18	18.22	4.00	Add	Math
AEX	•	•	•	0.18	43.4 + (4.0/char)	4.00	String Extract	ASCII
AHL	•	•	•	35.00	115.10	4.00	ASCII Handshake Lines	ASCII
AIC	•	•	•	0.18	110.00	2.00	Integer to String	ASCII
AND	•	•	•	0.56	1.50	3.00	And	Data Handling
ARD	•	•	•	35.00	151.00	3.00	ASCII Read Characters	ASCII
ARL	•	•	•	35.00	156.00	3.00	ASCII Read Line	ASCII
ASC	•	•	•	0.18	43.5 + (2.5/char)	4.00	String Search	ASCII
ASN① FP		•	•	0.56	41.45	2.00	Arc Sine	Math
ASR	•	•	•	0.18	43.50	3.00	ASCII String Compare	ASCII

Mnemonic FP = floating point	Applies to SLC 5/04 SLC 5/05 OS400	OS401	OS500	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
ATN① FP		•	•	0.56	40.15	2.00	Arc Tangent	Math
AWA	•	•	•	35.00	307.80	3.00	ASCII Write with Append	ASCII
AWT	•	•	•	35.00	217.30	3.00	ASCII Write	ASCII
BSL	•	•	•	7.50	31.5 + (2.31/word)	3.00	Bit Shift Left	Application Specific
BSR	•	•	•	7.50	31.5 + (2.31/word)	3.00	Bit Shift Right	Application Specific
CLR	•	•	•	0.56	1.50	3.00, 1.00	Clear	Math
CLR FP	•	•	•	0.18	5.94	1.00	Clear	Math
COP	•	•	•	0.56	20.2 + (2.0/word)	3.00	File Copy	Data Handling
COS① FP		•	•	0.56	37.20	2.00	Cosine	Math
CPT① FP		•	•	0.56	7.70②	③	Compute	Math
CTD	•	•	•	1.31	1.31	1.00	Count Down	Basic
CTU	•	•	•	1.31	1.31	1.00	Count Up	Basic
DCD	•	•	•	0.37	8.88	2.00	Decode 4 to 1 of 16	Data Handling
DDV	•	•	•	0.37	29.60	2.00	Double Divide	Math
DEG① FP		•	•	0.56	24.70	2.00	Degree	Data Handling
DIV	•	•	•	0.56	25.90	3.00, 4.00	Divide	Math
DIV FP	•	•	•	0.18	23.27	4.00	Divide	Math
EQU④	•	•	•	1.25	1.12	3.00	Equal	Comparison
EQU FP	•	•	•	0.56	12.50	3.00	Equal	Comparison
FFL	•	•	•	18.00	40.75	3.00	FIFO Load	Data Handling
FFU	•	•	•	18.00	60 + (2.0/word)	4.00	FIFO Unload	Data Handling
FLL	•	•	•	0.56	21.9 + (2.50/word)	3.00	Fill File	Data Handling
FRD	•	•	•	0.37	23.88	2.00	Convert from BCD	Data Handling
GEQ④	•	•	•	1.25	1.12	3.00	Greater Than or Equal	Comparison

Mnemonic FP = floating point	Applies to SLC 5/04 SLC 5/05 OS400	OS401	OS500	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
GEQ FP	•	•	•	0.56	14.31	3.00	Greater Than or Equal	Comparison
GRT④	•	•	•	1.25	1.12	3.00	Greater Than	Comparison
GRT FP	•	•	•	0.56	12.62	3.00	Greater Than	Comparison
IID	•	•	•	0.37	5.81	2.00	I/O Interrupt Disable	Interrupt
IIE	•	•	•	0.37	10.44	2.00	I/O Interrupt Enable	Interrupt
IIM⑤	•	•	•	0.37	51.00	6.00	Immediate Input with Mask	Program Flow Control
INT	•	•	•	0.18	0.18	1.00	Interrupt Subroutine	Application Specific
IOM⑤	•	•	•	0.37	75.74	6.00	Immediate Output with Mask	Program Flow Control
JMP	•	•	•	0.18	37.44	1.00	Jump to Label	Program Flow Control
JSR	•	•	•	0.18	112.00	1.00	Jump to Subroutine	Program Flow Control
LBL	•	•	•	0.18	0.18	1.00	Label	Program Flow Control
LEQ④	•	•	•	1.25	1.12	3.00	Less Than or Equal	Comparison
LEQ FP	•	•	•	0.56	13.93	3.00	Less Than or Equal	Comparison
LES④	•	•	•	1.25	1.12	3.00	Less Than	Comparison
LES FP	•	•	•	0.56	13.94	3.00	Less Than	Comparison
LFL	•	•	•	18.00	40.70	3.00	LIFO Load	Data Handling
LFU	•	•	•	18.00	34.70	3.00	LIFO Unload	Data Handling
LIM③	•	•	•	1.95	1.68	1.00	Limit Test	Comparison
LIM FP	•	•	•	0.56	20.19	1.00	Limit Test	Comparison
LN① FP		•	•	0.56	51.35	2.00	Natural Log	Math
LOG① FP		•	•	0.56	54.55	2.00	Log to the Base 10	Math
MCR	•	•	•	7.00	3.00	1.00	Master Control Reset	Program Flow Control

Mnemonic FP = floating point	Applies to SLC 5/04 SLC 5/05 OS400	OS401	OS500	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
MEQ④	•	•	•	NA	22.75	4.00	Masked Comparison for Equal	Comparison
MOV	•	•	•	0.37	1.12	2.00	Move	Data Handling
MOV FP	•	•	•	0.18	11.44	2.00	Move	Data Handling
MSG⑥	•	•	•	NA	183.00	20.00	Message	Communication
MUL	•	•	•	0.56	17.75	3.00, 4.00	Multiply	Math
MUL FP	•	•	•	0.18	21.94	3.00	Multiply	Math
MVM	•	•	•	0.56	17.40	3.00	Masked Move	Data Handling
NEG	•	•	•	0.56	1.50	3.00	Negate	Data Handling
NEG FP	•	•	•	0.18	11.87	3.00	Negate	Data Handling
NEQ④	•	•	•	1.25	1.12	3.00	Not Equal	Comparison
NEQ FP	•	•	•	0.56	12.18	3.00	Not Equal	Comparison
NOT	•	•	•	0.56	1.50	3.00	Not	Data Handling
OR	•	•	•	0.56	1.50	3.00	Or	Data Handling
OSR	•	•	•	5.75	9.10	2.00	One–Shot Rising	Basic
OTE	•	•	•	0.56	0.56	1.00	Output Energize	Basic
OTL	•	•	•	0.56	0.56	1.00	Output Latch	Basic
OTU	•	•	•	0.56	0.56	1.00	Output Unlatch	Basic
PID	•	•	•	14.31	169.82	26.00	Proportional Integral Derivative	PID
RAD① FP		•	•	0.56	24.65	2.00	Radian	Data Handling
REF	•	•	•	0.18	200⑦	1.00	I/O Refresh	Program Flow Control
RES	•	•	•	1.31	1.31	1.00	Reset	Basic
RET	•	•	•	0.18	20.00	1.00	Return from Subroutine	Program Flow Control
RPI	•	•	•	0.37	91 + (56/slot)	2.00	Reset Pending Interrupt	Interrupt
RTO	•	•	•	1.31	1.31	1.00	Retentive Timer	Basic

Mnemonic FP = floating point	Applies to SLC 5/04 SLC 5/05 OS400	OS401	OS500	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
SBR	•	•	•	0.18	0.18	1.00	Subroutine	Program Flow Control
SCL① FP		•	•	.748	33.06	4.00	Scale Data	Math
SCP	•	•	•	0.56	29.85	6.00	Scale with Parameters	Math
SCP FP		•	•	0.75	94.15	6.00	Scale with Parameters	Math
SIN① FP		•	•	0.56	38.05	2.00	Sine	Math
SQC	•	•	•	7.10	33.20	5.00	Sequencer Compare	Application Specific
SQL	•	•	•	7.10	33.20	4.00	Sequencer Load	Application Specific
SQO	•	•	•	7.10	44.10	5.00	Sequencer Output	Application Specific
SQR	•	•	•	0.37	28.80	2.00, 3.00	Square Root	Math
SQR FP	•	•	•	0.18	18.87	3.00	Square Root	Math
STD	•	•	•	0.18	3.56	1.00	Selectable Timer Interrupt Disable	Application Specific
STE	•	•	•	0.18	5.00	1.00	Selectable Timer Interrupt Enable	Application Specific
STS	•	•	•	0.56	44.38	3.00	Selectable Timer Interrupt Start	Application Specific
SUB	•	•	•	0.56	1.50	3.00, 4.00	Subtract	Math
SUB FP	•	•	•	0.18	19.50	4.00	Subtract	Math
SUS	•	•	•	0.37	10.31	2.00	Suspend	Program Flow Control
SVC	•	•	•	0.18	200⑧	1.00	Service Communication	Communication
SWP		•	•	0.56	22.6 + 12.13/word	2.00	SWAP	Math
TAN① FP		•	•	0.56	43.00	2.00	Tangent	Math
TND	•	•	•	NA	13.05	1.00	Temporary End	Program Flow Control
TOD	•	•	•	0.37	34.06	2.00	Convert to BCD	Data Handling

Mnemonic FP = floating point	Applies to SLC 5/04 SLC 5/05 OS400	OS401	OS500	False Execution Time (µs)	True Execution Time (µs)	Memory Usage (user words)	Name	Instruction Type
TOF	•	•	•	1.31	1.31	1.00	Timer Off–Delay	Basic
TON	•	•	•	1.31	1.31	1.00	Timer On–Delay	Basic
XIC④	•	•	•	0.37	0.37	1.00	Examine If Closed	Basic
XIO④	•	•	•	0.37	0.37	1.00	Examine If Open	Basic
XOR	•	•	•	0.56	1.50	3.00	Exclusive Or	Data Handling
XPY① FP		•	•	0.56	335.10	3.00	X to the Power of Y	Math

① The execution times assume floating point data. If signed integer data is used, add 15 microseconds per instruction execution time.

② To get the total execution time for a CPT instruction, take the CPT execution time plus each additional math instruction execution time, plus the number of math instructions times 3.01. For example, if a CPT instruction calls one ADD and one SUB instruction the calculation is: 7.7 + 1.70 + 1.70 + 2(3.01) = 17.12

③ To calculate the memory usage, do the following: Take 2 plus the number of instruction words for each operation performed plus the number of operations performed in the compute. For example, 2 + ADD + SUB + 2 =10.

④ These instructions take zero execution time if they are preceded by conditions that guarantee the state of the rung. Rung logic is solved left to right. Branches are solved from top to bottom.

⑤ Times listed are for discrete I/O modules. When using 32-point I/O modules, add the following microseconds to the IIM and IOM instructions:

- 15 µs IIM when true
- 30 µs IOM when true

When using the following modules and the IIM instruction in your program, add the following microseconds:

- Analog or Thermocouple module inputs, add 450 to 550 µs
- BASIC Module inputs, add 500 to 550 µs
- Other specialty inputs, add 425 to 957 µs

When using the following modules and the IOM instruction in your program, add the following microseconds:

- Analog or Thermocouple module inputs, add 390 to 416 µs
- BASIC Module inputs, add 440 to 466 µs
- Other specialty inputs, add 590 to 989 µs

⑥ This only includes the amount of time needed to “set up” the operation requested. It does not include the time it takes to service the actual communications.

⑦ This instruction performs a complete end of scan. This includes an input/output scan, service communication, and housekeeping. See Worksheet F in appendix D for calculating the actual execution time.

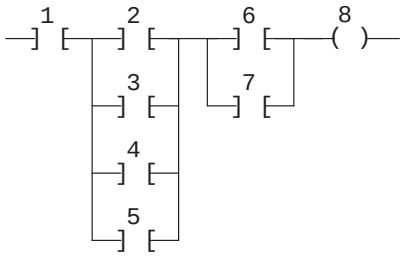
⑧ 1 channel = 150 µs no commands pending
2 channel = 170 µs no commands pending
Add 1 ms per command serviced.

SLC 5/04 or SLC 5/05 Execution Time Example

For the rung example below:

1) If instruction 1 is false, instructions 2, 3, 4, 5, 6, 7 take zero execution time. Execution time = .37 + .56 = .93 microseconds.

2) If instruction 1 is true, 2 is true, and 6 is true, then instructions 3, 4, 5, 7 take zero execution time. Execution time = .37 + .37 + .37 + .56 = 1.67 microseconds.



SLC 5/04 and SLC 5/05 Processor Floating Point Operations

Refer to the table on page C–34 for the instruction execution times when using floating point operations.

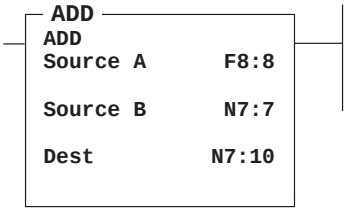
The floating point math times in the following table are applicable when S:34/2 is not set. When S:34/2 is set, the math flags are not updated after the instruction is executed and 4.8 μs is subtracted from the execution time. S:34/2 may be changed dynamically (that is, while the program is running, by the program). For each integer parameter, add 10 μs to allow for the integer to floating point conversion.

If you need to perform floating point operations using a mixture of floating point and integer parameters (values and source/destination addresses), calculate the amount of instruction words according to these guidelines:

1. Start with the amount of words for the floating point instruction.
2. Add 2 words for the first integer parameter (value or source/destination address).
3. Add 1 word for every subsequent integer parameter.

These additional words are required for integer-to-floating-point conversions.

For example, this instruction requires 7 instruction words:



- 4 words for the floating point ADD instruction (from the table on page C–41).
- + 2 words for the integer address, N7:7
- + 1 word for the second integer address, N7:10

Estimating Total Memory Usage of Your System Using an SLC 5/04 or SLC 5/05 Processor

_____	**1.**	Add the total number of data file words used (excluding the status file and I/O data words) and enter the result.
_____	**2.**	Multiply the total number of I/O data words by 3 and enter the result.
_____	**3.**	Multiply the total number of I/O slots, used or unused, by 3 and enter the result.
_____	**4.**	To account for processor overhead, enter 250 and enter the result.
_____	**5.**	Multiply the highest numbered data table file used by 5 and enter the result.
_____	**6.**	Multiply the highest numbered program file used by 5 and enter the result.
Subtotal: _____	**7.**	Add steps 1 through 6. Enter this as the subtotal (additional 4K word usage).
4096 – _____ (step 7)	**8.**	Subtract the value in step 7 from 4096; if the result is positive, enter 20,480 in step 14. If the result is negative, subtract the absolute value from 20,480 and enter the result in step 14. (This decreases the value.)
_____		
_____	**9.**	Calculate the total number of words used by the instructions in your program and enter the result. Refer to the table on page C–41.
_____	**10.**	Add the total number of rungs (1 word per rung) and enter the result.
_____	**11.**	Add 1 word for each indexed address reference and enter the result.
_____	**12.**	Add 2 words per rung for each rung that contains an indexed address reference and enter the result.
Subtotal: _____	**13.**	Add steps 9 through 12 and enter the result.
_____	**14.**	Enter the result from step 8. This is the available memory.
_____	**15.**	Enter the result from step 13. This is the total number of words used.
Total: _____	**16.**	Subtract step 15 from step 14. This number is the amount of memory available to your system.

SLC 5/04 and SLC 5/05 – Instructions Having Indexed Addresses

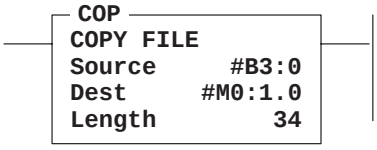
For each operand having an indexed address, add 25 µs microseconds to the execution time for a true instruction. For example, if a MOV instruction has an indexed address for both the source and destination, the execution time when the instruction is true is 1.12 + 25 + 25 = 51.12 microseconds.

SLC 5/04 and SLC 5/05 – Instructions Having M0 and M1 Data File Addresses

Instructions having M0 and M1 data file addresses vary in their execution times. The following execution times represent the expected maximum values.

Instruction Type	Execution Time (µs)
XIC or XIO	743
OTU, OTE, or OTL	879
COP to M file	735 + 23 per word
COP from M file	722 + 22 per word
FLL	716 + 30 per word
MVM to M file	850
any Source or Destination M file address	694

Example



For the multi–word instruction above, add 735 microseconds plus 23 microseconds per word. In this example, 34 words are copied from #B3:0 to M0:1.0. Add 735 + (23 x 34) = 1517 microseconds to the execution time listed on page C–35. This comes to 88.54 + 1517 = 1605.5 microseconds total, or 1.6 milliseconds.

Indirect Addressing

The following sections describe how indirect addressing affects the execution time of instructions in the SLC 5/03 OS302, SLC 5/04 OS401, and SLC 5/05 processors. The timing for an indirect address is affected by:

- the form of the indirect address
- whether the indirect address is a source or destination parameter
- whether indirect addressing is used in either a COP, FLL, FFL/FFU, LFL/LFU, BSR, BSL, or MVM instruction
- whether indirect addressing is used in either an XIC, XIO, OTU, OTL, OTE, or OSR instruction

For the address forms in the table on the next page, you can substitute the following file types:

For an Integer (N)	For a String (ST)
Input (I)	Control (R)
Output (O)	Counter (C)
Bit (B)	Timer (T)
Floating Point (F)	
ASCII (A)	

Execution Times for Word-Level Indirect Addresses

For most types of instructions that contain an indirect address(es), look up the form of the indirect address in the table below and add that time to the execution time of the instruction.

Address Form①	Source Operand (μs)		Destination Operand (μs)		If used in a file type instruction	
	SLC 5/03	SLC 5/04 SLC 5/05	SLC 5/03	SLC 5/04 SLC 5/05	SLC 5/03	SLC 5/04 SLC 5/05
N7:[*]	65.1	56.15	63.10	54.20	76.35	66.75
ST12:[*].[*]	69.45	60.00	67.45	58.05	80.70	70.60
ST12:[*].0	74.65	59.60	72.65	57.65	85.90	70.20
ST12:0.[*]	74.65	59.60	72.65	57.65	85.90	70.20
N[*]:[*]	105.90	89.40	131.50	112.55	138.75	118.70
N[*]:0	111.10	89.00	136.70	112.15	143.95	118.30
ST[*]:[*].[*]	110.25	93.25	135.85	116.40	143.10	122.55
ST[*]:[*].0	115.45	92.85	141.05	116.00	148.30	122.15
ST[*]:0.[*]	115.45	92.85	141.05	116.00	148.30	122.15
ST[*]:0.0	120.65	92.45	146.25	115.60	153.50	121.75
#N7:[*]	73.05	59.35	64.65	57.30	86.80	69.80
#ST12:[*].[*]	77.40	63.20	69.00	61.15	91.15	73.65
#ST12:[*].0	82.60	62.80	74.20	60.75	96.35	73.25
#ST12:0.[*]	82.60	62.80	74.20	60.75	96.35	73.25
#N[*]:[*]	110.95	92.95	133.40	114.40	146.65	121.35
#N[*]:0	116.15	92.55	138.60	114.00	151.85	120.95
#ST[*]:[*].[*]	115.30	96.80	137.75	118.25	151.00	125.20
#ST[*]:[*].0	120.50	96.40	142.95	117.85	156.20	124.80
#ST[*]:0.[*]	120.50	96.40	142.95	117.85	156.20	124.80
#ST[*]:0.0	125.70	96.00	148.15	117.45	161.40	124.40

① [*] indicates that an indirect reference is substituted.

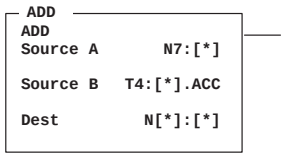
Execution Times for Bit-Level Indirect Addresses

Indirect bit addresses are based on the form of the indirect address and the type of bit instruction. Use the following two tables to calculate the execution time of a bit instruction.

Address Form	Additional Time (μs)	
	SLC 5/03	SLC 5/04 SLC 5/05
B3/[*]	96.70	77.80
B3:1/[*]	96.70	77.80
B3:[*]/[*]	91.50	72.80
ST12:[*].[*]/[*]	100.65	76.65
ST12:[*].[*]/0	100.85	76.25
ST12:[*].0/[*]	100.85	76.25
ST12:[*].0/0	105.85	75.85
ST12:0.[*] /0	105.85	75.85
ST12:0.0/[*]	105.85	75.85
B[*]/[*]	171.50	141.40
B[*]:1/[*]	171.50	141.40
B[*]:[*]/[*]	166.30	141.80
ST[*]:[*].[*]/[*]	170.65	145.65
ST[*]:[*].[*]/0	175.85	145.25
ST[*]:[*].0/[*]	175.85	145.25
ST[*]:[*].0/0	181.05	144.85
ST[*]:0.[*]/[*]	175.85	145.25
ST[*]:0.[*]/0	181.05	144.85
ST[*]:0.0/[*]	181.05	144.85
ST[*]:0.0/0	186.25	144.45

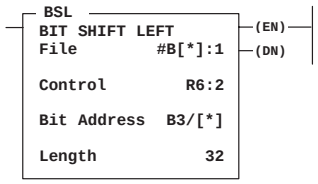
Execution Time Examples – Word Level and Bit Level Indirect Address

SLC 5/03



ADD 1.70
Source A 65.10
Source B 74.65
Destination 131.50
272.95 µs

SLC 5/04
SLC 5/05



BSL . . . . . 31.5 + (2)2.31= 36.12
File 120.95
Bit Address 77.80
234.87 µs

Instruction Execution Times

Instruction	Execution Time (µs)	
	SLC 5/03	SLC 5/04 SLC 5/05
XIC	10.20	8.72
XIO	14.65	12.76
OTU	6.30	5.45
OTL	9.35	5.40
OTE	6.25	5.50
OSR	10.50	8.10

Execution Time Example – Bit Instruction Using an Indirect Address

To calculate the execution time of an XIC at B3/[N7:0] using an SLC 5/03 processor add the following:

Execution Time Using Indirect Address =
Execution Time for Bit-Level Indirect Address + Instruction Execution Time =
10.20 + 96.70 = 106.90

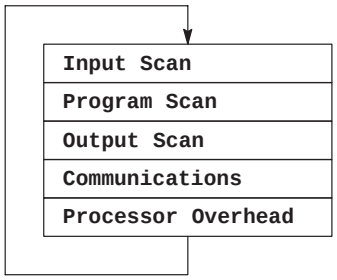
D Estimating Scan Time

This appendix contains worksheets that allow you to estimate the scan time for your particular controller configuration and program. Scan time calculation for an example controller and program are included to help you.

You will be using the instruction execution times listed in appendix C.

Processor Operating Cycle

The diagram and table below breaks down the processor operating cycle into events. Directions for calculating the scan time of these events appear in the worksheets.



Events in the processor operating cycle

Event	Description
Input Scan	The status of input modules is read and the input image in the processor is updated with this information.
Program Scan	The ladder program is executed. The input image table is evaluated, ladder rungs are solved, and the output image is updated. The information is not yet transferred to the output modules.
Output Scan	The output image information is transferred to the output modules.
Communications	Communication with programmers and other network devices takes place.
Processor Overhead	Processor internal housekeeping takes place. Actions include performing program pre–scan and updating the internal timebase and the Status file.

Access Times for M0/M1 Data

During the program scan, the processor must access the specialty I/O card to read/write M0 or M1 data. This access time must be added to the execution time of each instruction referencing M0 or M1 data. The following table shows where to find the approximate access times per instruction or word of data.

Processor	Access Time per Bit Instruction or Word of Data	Access Time per Multi–Word Instruction
SLC 5/02	See page C–20.	See page C–20.
SLC 5/03	See page C–33.	See page C–33.
SLC 5/04	See page C–43.	See page C–43.
SLC 5/05	See page C–43.	See page C–43.

Interrupt Latency

MicroLogix 1000 User Interrupt Latency

The user interrupt latency is the maximum time from when an interrupt condition occurs (e.g., STI expires or HSC preset is reached) to when the user interrupt subroutine begins executing (assumes that there are no other interrupt conditions present).

If you are communicating with the controller, the maximum user interrupt latency is 872 µs. If you are *not* communicating with the controller, the maximum user interrupt latency is 838 µs.

SLC User Interrupt Latency

Interrupt latency is the interval between interrupt detection and the start of the interrupt subroutine. During this time frame, the SLC 500 processor performs operations that cannot be interrupted.

Note *If you are using a SLC 5/03 processor and S:33/8 is reset (0), interrupts* may not *be serviced within the calculated interrupt latency period. (See next page.) This applies to the following instructions.*

- Selectable Timed Interrupt (STI)
- Discrete Input Interrupt (DII)
- I/O Interrupt

Calculating Interrupt Latency for SLC 5/03

These examples assume S:33/8 Interrupt Latency Control bit is set.

OS302 FRN 9 and lower	OS302 FRN 10 and higher	
		Selectable Timed Interrupt (STI)
288 µs	425 µs	Entry Time
60 µs	60 µs	Fixed Overhead
		Channel 0 Background Communication Overhead
		Channel 1 Background Communication Overhead
		Total Interrupt Latency (max.)
		Discrete Input Interrupt (DII)
326 µs	440 µs	Entry Time
60 µs	60 µs	Fixed Overhead
		Channel 0 Background Communication Overhead
		Channel 1 Background Communication Overhead
		Total Interrupt Latency (max.)
		I/O Event Interrupt (IOI)
608 µs	715 µs	Entry Time
60 µs	60 µs	Fixed Overhead
		Channel 0 Background Communication Overhead
		Channel 1 Background Communication Overhead
		Total Interrupt Latency (max.)

Use the following execution times to help calculate your interrupt latency.

Channel Configuration	Background Overhead Time
DH–485	300 µs
DF1 Half–Duplex Master/Slave	650 µs①
DF1 Full–Duplex	655 µs①

① The above times include hardware handshaking. These times also assume that secure modem handshaking exists. If the modem lines are noisy, add 80 µs to the specific time.

Calculating Interrupt Latency for SLC 5/04 or SLC 5/05

These examples assume S:33/8 Interrupt Latency Control bit is set.

OS401 FRN 6 and lower	OS401 FRN 7 and higher	
		Selectable Timed Interrupt (STI)
237 µs	355 µs	Entry Time
60 µs	60 µs	Fixed Overhead
_____	_____	Channel 0 Background Communication Overhead
_____	_____	Channel 1 Background Communication Overhead
_____	_____	**Total Interrupt Latency (max.)**
		Discrete Input Interrupt (DII)
278 µs	365 µs	Entry Time
60 µs	60 µs	Fixed Overhead
_____	_____	Channel 0 Background Communication Overhead
_____	_____	Channel 1 Background Communication Overhead
_____	_____	**Total Interrupt Latency (max.)**
		I/O Event Interrupt (IOI)
472 µs	595 µs	Entry Time
60 µs	60 µs	Fixed Overhead
_____	_____	Channel 0 Background Communication Overhead
_____	_____	Channel 1 Background Communication Overhead
_____	_____	**Total Interrupt Latency (max.)**

Use the following execution times to help calculate your interrupt latency.

Channel Configuration	Background Overhead Time
DH–485	220 µs
DF1 Half–Duplex Master/Slave	553 µs①
DF1 Full–Duplex	553 µs①
DH+	260 µs
Ethernet	260 µs

① The above times include hardware handshaking. These times also assume that secure modem handshaking exists. If the modem lines are noisy, add 80 µs to the specific time.

Example – SLC 5/03 (OS302, FRN10 and higher) Processor Selectable Timed Interrupt

Configuration: Channel 0 shutdown
Channel 1 DH–485

425 µs	Entry Time
60 µs	Fixed Overhead
0 µs	Channel 0 Background Communication Overhead
300 µs	Channel 1 Background Communication Overhead
785 µs	**Total Interrupt Latency (max.)**

Example – SLC 5/05 Processor Selectable Timed Interrupt

Configuration: Channel 0 DH–485
Channel 1 Ethernet

355 µs	Entry Time
60 µs	Fixed Overhead
220 µs	Channel 0 Background Communication Overhead
260 µs	Channel 1 Background Communication Overhead
895 µs	**Total Interrupt Latency (max.)**

Scan Time Worksheets

Worksheets on the following pages are as follows:

An example scan time calculation appears on page D–24.

These worksheets are intended to assist you in estimating scan time for your application. Refer to appendix C for instruction execution times. Refer to the *SLC 500 Family System Overview,* Publication 1747–2.30, for I/O module part numbers and sizes.

Defining Worksheet Terminology

When you work through the worksheets, you will come across the following terms:

Term:	Definition:
Background Communication	Occurs when your processor is attached to an active network. During this event the processor accepts characters from the network and places them into a packet buffer.
Foreground Communication	Occurs only when another node is attached, or when another processor sends an MSG instruction to your processor. During this event the processor performs the communication commands contained in completed packets built during background communications.
Forced Input Overhead	This value is included in your scan time whenever forces are Enabled in your program.
Forced Output Overhead	This value is included in your scan time whenever forces are Enabled in your program.
Processor Overhead Times	Processor internal housekeeping takes place. Actions include performing program pre–scan and updating the internal timebase and the Status file.
Single Step	When using this function with a SLC 5/02, SLC 5/03, SLC 5/04, or SLC 5/05 processor, you can execute your program one rung or section at a time. This function is used for debugging purposes.
Multi–word Module	Example of multi–word modules are DCM, analog, and DSN.

Worksheet A – Estimating the Scan Time of Your MicroLogix 1000 Controller

Use this worksheet to calculate your execution time for ladder program.

Procedure		Maximum Scan Time
1.	**Input scan time, output scan time, housekeeping time, and forcing.**	210 µs (discrete) 330 µs with forcing (analog) 250 µs without forcing (analog)
2.	**Estimate your program scan time:**	
	A. Count the number of program rungs in your logic program and multiply by 6.	_____ µs
	B. Add up your program execution times when all instructions are true. Include interrupt routines in this calculation.[1]	_____ µs
3.	**Estimate your controller scan time:**	
	A. Without communications, add sections 1 and 2	_____ µs
	B. With communications, add sections 1 and 2 and multiply by 1.05	_____ µs
4.	**To determine your maximum scan time in ms, divide your controller scan time by 1000.**	_____ ms

[1] If a subroutine executes more than once per scan, include each subroutine execution scan time.

Worksheet B – Estimating the Scan Time of Your Fixed Controller

Procedure	Min Scan Time	Max Scan Time
1. Estimate your *input* scan time (µs).		
1.1 Determine the type of controller that you have. If you have a 20 I/O processor, write 313 on line (A). If you have a 30 or 40 I/O processor, write 429 on line (A). A.)_____		
1.2 Calculate the processor input scan of your discrete input modules (this is a zero if not using the expansion chassis) Number of 8 point modules _____ x 197 = B.)_____ Number of 16 point modules _____ x 313 = C.)_____ Number of 32 point modules _____ x 545 = D.)_____		
1.3 Calculate the processor input scan of your specialty I/O modules (this is a zero if not using the expansion chassis) Number of 1/4 DCM or analog combo _____ x 652 = E.)_____ Number of 1/2 DCM, analog input, or 1746–HS _____ x 1126 = F.) _____ Number of 3/4 DCM _____ x 1600 = G.)_____ Number of full DCM, BASIC, or 1747–DSN _____ x 2076 = H.)_____ Number of 1747–KE _____ x 443 = I.) _____ Number of 1746–NT4 _____ x = J.) _____		
1.4 Add lines A through J. Place this value on line (K). Add 101 to the value on line (K). This sum is your *minimum* input scan time. K.)_____ + 101 = L	**L)**_____	
1.5 Calculate your *maximum* input scan time: Maximum input scan time (M) = Minimum input scan time (K) + (Number of specialty I/O modules x 50)		**M)**_____
1.6 Calculate the Forced Input Overhead: Forced Input Overhead = (N) (Number of input modules x 180) + 140 per additional word for multi–word modules (e.g. DCM, analog, DSN) When forces are disabled this value is 0.	**N)**_____	**N)**_____

Continued on next page

Procedure	Min Scan Time	Max Scan Time
2. Estimate your *output* scan time (µs). 2.1 Determine the type of controller that you have. If you have a 20 I/O processor, write 173 on line (A). If you have a 30 or 40 I/O processor, write 272 on line (A). A.)_____ 2.2 Calculate the processor output scan of your discrete output modules. Number of 8 point modules _____ x 173 = B.)_____ Number of 16 point modules _____ x 272 = C.)_____ Number of 32 point modules _____ x 470 = D.)_____ 2.3 Calculate the processor output scan of your specialty I/O modules. Number of 1/4 DCM or analog combo _____ x 620 = E.)_____ Number of 1/2 DCM, analog output, or 1746–HS _____ x 1028 = F.) _____ Number of 3/4 DCM _____ x 1436 = G.)_____ Number of full DCM, BASIC, or 1747–DSN _____ x 1844 = H.)_____ 2.4 Add lines A through H. Place this value on line (I). Add 129 to the value on line (I). This sum is your *minimum* output scan time. I.)_____ + 129 = (J)	**J)**_____	
2.5 Calculate your *maximum* output scan time: Maximum output scan time (K) = Minimum scan time (J) + (Number of specialty I/O modules x 50)		**K)**_____
2.6 Calculate the Forced Output Overhead: Forced Output Overhead = (L) (Number of output modules x 172) + 140 per additional word for multi–word modules (e.g. DCM, analog, DSN) When forces are disabled this value is 0.	**L)**_____	**L)**_____
3. Estimate your *program* scan time. This estimate assumes operation of all instructions once per operating scan. 3.1 Count the number of rungs in your program. Place value on line (A).	**A)**_____	**A)**_____
3.2 Calculate your maximum program execution time (B) when all instructions are true.		**B)**_____
3.3 Calculate the minimum program execution time (C) using the times associated with an instruction when false. (See appendix C to do this.)	**C)**_____	
4. Add the values in the minimum and maximum scan time columns.	_____ subtotal	_____ subtotal
5. Add *processor* overhead time (178 for min. scan time; 278 for max. scan time) to the subtotals estimated in step 4. Use these new subtotals to calculate communication overhead in step 6.	+ 178 _____ subtotal	+ 278 _____ subtotal
6. Estimate your *communication* overhead: 6.1 Calculate the background communication overhead when no communication exist: multiply the subtotal for minimum scan time (A) (estimated in step 5) by 1; multiply the subtotal for maximum scan time (B) by 1.140 (max. value accounts for active DH–485 link) when communication exists.	**A** x 1.000 _____ µsecs.	**B** x 1.140 _____ µsecs.
6.2 Calculate the foreground communication overhead: for minimum scan time (C) add 0; for maximum scan time (D) add 2310. (Maximum scan time accounts for programmer being attached to processor.)	**C** + 0	**D** + 2310
6.3 Convert µsecs. to msecs., divide by 1000.	_____ µsecs. / 1000	_____ µsecs. / 1000
Estimated minimum and maximum scan times for your fixed controller application:	msecs.	msecs.

Worksheet C – Estimating the Scan Time of Your SLC 5/01 Processor

Procedure	Min Scan Time	Max Scan Time
1. Estimate your *input* scan time (µs).		
1.1 Calculate the processor input scan of your discrete input modules.		
Number of 8 point modules _____ x 197 = A.)_____		
Number of 16 point modules _____ x 313 = B.)_____		
Number of 32 point modules _____ x 545 = C.)_____		
1.2 Calculate the processor input scan of your specialty I/O modules.		
Number of 1/4 DCM or analog combo _____ x 652 = D.)_____		
Number of 1/2 DCM, analog input, 1746–HS _____ x 1126 = E.)_____		
Number of 3/4 DCM _____ x 1600 = F)_____		
Number of full DCM, BASIC, or 1747–DSN _____ x 2076 = G.)_____		
Number of 1747–KE _____ x 443 = H.)_____		
Number of 1746–NT4 _____ x = I.)_____		
1.3 Add lines A through I. Place this value on line (J) Add 101 to the value on line (J). This sum is your *minimum* input scan time. J.)_____ + 101 = K	**K)**_____	
1.4 Calculate your *maximum* input scan time: Maximum input scan time (L) = Minimum scan time (K) + (Number of specialty I/O modules x 50)		**L)**_____
1.5 Calculate the Forced Input Overhead: Forced Input Overhead = (M) (Number of input modules x 180) + 140 per additional word for multi–word modules (e.g. DCM, analog, DSN)		**M)**_____
2. Estimate your *output* scan time (µs).		
2.1 Calculate the processor output scan of your discrete output modules.		
Number of 8 point modules _____ x 173 = A.)_____		
Number of 16 point modules _____ x 272 = B.)_____		
Number of 32 point modules _____ x 470 = C.)_____		
2.2 Calculate the processor output scan of your specialty I/O modules.		
Number of 1/4 DCM or analog combo _____ x 620 = D.)_____		
Number of 1/2 DCM, analog output, or 1746–HS _____ x 1028 = E.)_____		
Number of 3/4 DCM _____ x 1436 = F) _____		
Number of full DCM, BASIC, or 1747–DSN _____ x 1844 = G.)_____		
2.3 Add lines A through G. Place this value on line (H). Add 129 to the value on line (H). This sum is your *minimum* output scan time. H)_____ + 129 = I	**I)**_____	
2.4 Calculate your *maximum* output scan time: Maximum output scan time (J) = Minimum scan time (I) + (Number of specialty I/O modules x 50)		**J)**_____
2.5 Calculate the Forced Output Overhead: Forced Output Overhead = K (Number of output modules x 172) + 140 per additional word for multi–word modules (e.g. DCM, analog, DSN)		**K)**_____

Continued on next page

Procedure	Min Scan Time	Max Scan Time
3. Estimate your *program* scan time. This estimate assumes operation of all instructions once per operating scan. 3.1 Count the number of rungs in your program. Place value on line (A). 3.2 Calculate your program execution time (B) when all instructions are true. (See appendix C to do this.) **4. Add the values in the minimum and maximum scan time columns.**	**A)**_____ **B)**_____ _____ subtotal	**A)**_____ **B)**_____ _____ subtotal
5. Add *processor* overhead time (178 for min. scan time; 278 for max. scan time) to the subtotals estimated in step 4. Use these new subtotals to calculate communications overhead in step 6.	+ 178 _____ subtotal	+ 278 _____ subtotal
6. Estimate your *communication* overhead: 6.1 Calculate the background communication overhead: multiply the subtotal for minimum scan time (estimated in step 5) by 1; multiply the subtotal for maximum scan time by 1.140 (max. value accounts for active DH–485 link). 6.2 Calculate the foreground communication overhead: for minimum scan time add 0; for maximum scan time add 2310. (Maximum scan time accounts for programmer being attached to processor.) 6.3 Convert µsecs. to msecs., divide by 1000.	x 1.000 _____ µsecs. + 0 _____ µsecs. / 1000	x 1.140 _____ µsecs. + 2310 _____ µsecs. / 1000
Estimated minimum and maximum scan times for your SLC 5/01 processor application:	msecs.	msecs.

Worksheet D – Estimating the Scan Time of Your SLC 5/02 Processor

Procedure	Min Scan Time	Max Scan Time
1. Estimate your *input* scan time (µs). 1.1 Calculate the processor input scan of your discrete input modules. Number of 8 point modules _____ x 126 = A.)_____ Number of 16 point modules _____ x 195 = B.)_____ Number of 32 point modules _____ x 335 = C.)_____ 1.2 Calculate the processor input scan of your specialty I/O modules. Number of 1/4 DCM or analog combo _____ x 375 = D.)_____ Number of 1/2 DCM, analog input, 1746–HS _____ x 659 = E.)_____ Number of 3/4 DCM _____ x 944 = F)_____ No. of full DCM, BASIC small config., or 7–block DSN _____ x 1228 = G.)_____ Number of 1747–KE _____ x 250 = H.)_____ Number of 1746–NT4 _____ x = I.)_____ 1.3 Calculate the processor input scan of your specialty I/O modules. Number of BASIC Lg. config., 1746–HSCE _____ x 1557 = J.)_____ Number of RI/O Scanner or 30–block DSN _____ x 4970 = K.)_____ 1.4 Add lines A through K. Place this value on line (L). Add 121 to the value on line (L). This sum is your *minimum* input scan time. L.)_____ + 121 = M	**M)**_____	
1.5 Calculate the *maximum* input scan time: Minimum scan time (M) + (Number of specialty I/O modules in section 1.2 x 30) + (Number of specialty I/O modules in section 1.3 x 120) = (N)		**N)**_____
1.6 Calculate Forced Input Overhead (O) = (No. of input modules x 108) + 140 per additional word for multi–word modules		**O)**_____

Continued on next page

Procedure	Min Scan Time	Max Scan Time
2. Estimate your *output* scan time (µs). 2.1 Calculate the processor output scan of your discrete output modules. Number of 8 point modules _____ x 104 = A.)_____ Number of 16 point modules _____ x 164 = B.)_____ Number of 32 point modules _____ x 282 = C.)_____ 2.2 Calculate the processor output scan of your specialty I/O modules. Number of 1/4 DCM or analog combo _____ x 372 = D.)_____ Number of 1/2 DCM, analog output, 1746–HS _____ x 617 = E.)_____ Number of 3/4 DCM _____ x 862 = F.)_____ No. of full DCM, BASIC small config., or 7–block DSN _____ x 1047 = G.)_____ 2.3 Calculate the processor output scan of your specialty I/O modules. Number of BASIC Lg. config., 1746–HSCE _____ x 1399 = H.)_____ Number of RI/O Scanner or 30–block DSN _____ x 4367 = I.)_____ 2.4 Add lines A through I. Place this value on line (J). Add 138 to the value on line (J). This sum is your *minimum* output scan time. J.)_____ + 138 = K	**K)**_____	
2.5 Calculate your *maximum* output scan time = L Minimum scan time (K) + (Number of specialty I/O modules in part B x 30) + (Number of specialty I/O modules in part C x 120)		**L)**_____
2.6 Calculate the Forced Output Overhead (M)= (No. of output modules x 104) + 140 per additional word for multi–word modules		**M)**_____
3. Estimate your *program* scan time. This estimate assumes operation of all instructions once per operating scan. 3.1 Count the number of rungs in your program. Place value on line (A).	**A)**_____	**A)**_____
3.2 Multiply value on line (A) by 6. (If you saved your program with Single–Step Enabled, then multiply the value on line (A) by 66.) A.)_____ x 6 =	**B)**_____ _____ subtotal	_____ _____ subtotal
3.3 Calculate your program execution time when all instructions are true. (See appendix C to do this.)		
4. Add the values in the minimum and maximum scan time columns.		
5. Add *processor* overhead time (180 for min. scan time; 280 for max. scan time) to the subtotals estimated in step 4. Use these new subtotals to calculate communication overhead in step 6.	+ 180 _____subtotal	+ 280 _____ subtotal

Continued on next page

Procedure	Min Scan Time	Max Scan Time
6. Estimate your *communication* overhead: 6.1 Calculate the background communication overhead: multiply the subtotal for minimum scan time (estimated in step 5) by 1040; multiply the subtotal for maximum scan time by 1140 (max. value accounts for active DH–485 link). 6.2 Calculate the foreground communications overhead: for minimum scan time add 0; for maximum scan time add 2286. (Maximum scan time accounts for programmer being attached to processor.) 6.3 Convert µsecs. to msecs., divide by 1000.	x 1040 _____ µsecs. + 0 _____ µsecs. / 1000	x 1140 _____ µsecs. + 2286 _____ µsecs. / 1000
Estimated minimum and maximum scan times for your SLC 5/02 Series C application:	msecs.	msecs.
7. Estimate the scan time for your SLC 5/02 Series B processor application; divide the values for Series C application by 0.60. ***Estimated minimum and maximum scan times for your SLC 5/02 Series B processor application:***	/ 0.60 msecs.	/ 0.60 msecs.

Worksheet E – Estimating the Scan Time of Your SLC 5/03 Processor

Procedure	Min Scan Time	Max Scan Time
1. Estimate your *input* scan time (µs).		
1.1 Calculate the processor input scan of your discrete input modules.		
Number of 8 point modules _____ x 26 = A.)_____		
Number of 16 point modules _____ x 26 = B.)_____		
Number of 32 point modules _____ x 52 = C.)_____		
Number of I/O Combo modules _____ x 74 = D.)_____		
1.2 Calculate the processor input scan of your specialty I/O modules.		
Number of 1746–HS _____ x 332 = E.)_____		
Number of 1/4 DCM _____ x 317 = F.)_____		
Number of 1/2 DCM _____ x 352 = G.)_____		
Number of 3/4 DCM _____ x 478 = H.)_____		
Number of full DCM _____ x 420 = I)_____		
Number of 1747–KE _____ x 443 = J)_____		
Number of 1746–NI4 _____ x 316 = K.)_____		
Number of 1746–NIO4I, NIO4V _____ x 272 = L.)_____		
Number of 1746–NT4 _____ x 385 = M.)_____		
Number of 7–block DSN _____ x 423 = N.)_____		
Number of 30–block DSN _____ x 1051 = O.)_____		
Number of 1746–BAS (SLC 5/01 configuration) _____ x 451 = P.)_____		
1.3 Number of 1747–SN _____ x 1218 = Q.)_____		
Number of 1746–HSCE _____ x 506 = R.)_____		
Number of 1746–BAS (SLC 5/02 & SLC 5/03 configuration) _____ x 605 = S.)_____		
1.4 If S:33/8 is clear, add 92 µs per configured input slot _____ x 92 = T.)_____		
1.5 Add lines A through T. Place this value on line (W). Add 31 to value on line (W). This is the **minimum input scan.** W.) _____ + 31 = X	**X)**_____	**X)**_____
1.6 Calculate your maximum specialty I/O input scan time: Input scan time (R1) = Number of specialty I/O modules in section 1.2 x 50		**R1)**_____
1.7 Calculate your maximum I/O input scan time for section 1.3: Input scan time (R2) = Number of specialty I/O modules in section 1.3 x 200		**R2)**_____
1.8 Calculate the Forced Input Overhead: Forced Input Overhead = Number of discrete input modules at 3.5 µs per word Add 20 + 4/per word for each specialty I/O module (i.e. BASIC has 8 inputs = 20 + 4 x 8 = 52) = Y	**Y)**_____	**Y)**_____
1.9.0 Add the values in the minimum scan time columns; put the values in the minimum scan time subtotal.	**subtotal**_____	
1.9.1 Add the values in the maximum scan time columns; put the values in the maximum scan time subtotal.		**subtotal**_____

Continued on next page

Procedure	Min Scan Time	Max Scan Time
2. Estimate your *output* scan time (µs).		
2.1 Calculate the processor output scan of your discrete output modules.		
Number of 8 point modules _____ x 31 = A.)_____		
Number of 16 point modules _____ x 31 = B.)_____		
Number of 32 point modules _____ x 62 = C.)_____		
Number of I/O Combo modules _____ x 82 = D.)_____		
2.2 Calculate the processor output scan of your specialty I/O modules.		
Number of 1746–HS _____ x 369 = E.)_____		
Number of 1/4 DCM _____ x 335 = F.)_____		
Number of 1/2 DCM _____ x 380 = G.)_____		
Number of 3/4 DCM _____ x 424 = H.)_____		
Number of full DCM _____ x 469 = I.)_____		
Number of 1746–NIO4I & NIO4V _____ x 297 = J.)_____		
Number of 1746–NO4I & NO4V _____ x 342 = K.)_____		
Number of 7–block DSN _____ x 469 = L.)_____		
Number of 30–block DSN _____ x 1224 = M.)_____		
Number of 1746–BAS (SLC 5/01 configuration) _____ x 500 = O.)_____		
2.3 Calculate the processor output scan of your specialty I/O modules.		
Number of 1747–SN _____ x 1395 = P.)_____		
Number of 1746–HSCE _____ x 394 = Q.)_____		
Number of 1746–BAS (SLC 5/02 & SLC 5/03 configuration) _____ x 656 = R.)_____		
2.4 If S:33/8 is clear, add 92 µs per configured output slot _____ x 92 = S.)_____		
2.5 Add lines A through S. Place this value on line (T). Add 30 to value on line (T). This is your **minimum output** scan of your specialty I/O. T.)_____ + 30 =X	**X)**_____	**X)**_____
2.6 Calculate your maximum specialty output scan time: Output scan time (R1) = Number of specialty I/O modules in 2.2 x 50)		**R1)**_____
2.7 Calculate your maximum output scan time for section 2.3: Output scan (R2) = Number of specialty I/O modules x 200		**R2)**_____
2.8 Calculate the Forced Output Overhead: Forced Output Overhead = Number of discrete output modules at 3.0 µs per word Add 20 + 4/per word for each specialty I/O module (i.e. BASIC has 8 input words = 20 + 4 x 8 = 52)	**Y)**_____	**Y)**_____
2.9.0 Add the values in the minimum scan time columns; put the values in the minimum scan time subtotal.	**subtotal**_____	
2.9.1 Add the values in the maximum scan time columns; put the values in the maximum scan time subtotal.		**subtotal**_____

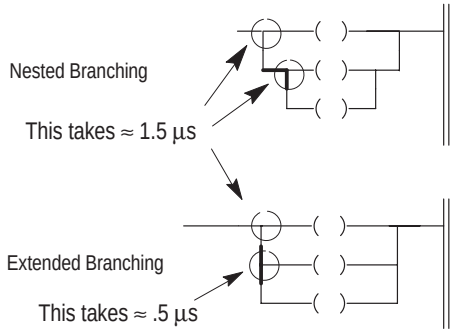
Continued on next page

Procedure	Min Scan Time	Max Scan Time
3. Estimate your *program* scan time. This estimate assumes operation of all instructions once per operating scan. 3.1 Count the number of rungs in your ladder program. Multiply this number by 0.25. Place the value on line Rungs. 3.2 Calculate your program execution time when all instructions are true. (See appendix C to do this.) Place this value on line Time. 3.3 Add the values from 3.1 and 3.2; place this on the subtotal line.	**Rungs)_____** **+** **Time)_____** **subtotal_____**	**Rungs)_____** **+** **Time)_____** **subtotal_____**
4. *Processor* overhead time (S:33/8 = 1 add 752 µs; S:33/8 = 0 add 844 µs)	**_____subtotal**	**_____subtotal**
5. Estimate your *communication* overhead: 5.1 Calculate the background communications overhead using the table below. The minimum overhead applies when devices *are not* connected to a channel.		

Channel 0 (RS232)	Channel 1 (DH485)	Background Communication Overhead	
		Minimum	Maximum
DF1 Full–Duplex	DH–485	1026	1180 (1280)
DF1 Half–Duplex	DH–485	1025	1175
DH–485	DH–485	1040	1160
DF1 Full–Duplex	Shutdown	1006	1100 (1200)
DF1 Half–Duplex	Shutdown	1005	1095
DH–485	Shutdown	1020	1080

Use the numbers in parentheses when using MSG instructions on DF1 Full–Duplex channel.	_____ Overhead	_____ Overhead
5.2 Calculate the foreground communications overhead: for minimum scan time add 0; for maximum scan time add 1027 per channel. (Maximum scan time accounts for programmer being attached to processor.)	+ **0** **_____subtotal**	+ **× 1027** **_____subtotal**
6. Total all the values from the subtotals of steps 1–5 (minimum and maximum input scan, output scan, program scan, processor overhead, and communication overhead).	_____µsecs.	_____µsecs.
7. Convert µsecs. to msecs. by dividing by 1000.	/ 1000	/ 1000
Estimated minimum and maximum scan times for your SLC 5/03 processor application:	msecs.	msecs.

Note *The above scan time does not account for branching. See the example below for how to estimate additional scan time when using branching.*



Worksheet F – Estimating Scan Time of Your SLC 5/04 or 5/05 Processor

Procedure	Min Scan Time	Max Scan Time
1. Estimate your *input* scan time (µs).		
1.1 Calculate the processor input scan of your discrete input modules.		
Number of 8 point modules _____ x 26 = A.)_____		
Number of 16 point modules _____ x 26 = B.)_____		
Number of 32 point modules _____ x 52 = C.)_____		
Number of I/O Combo modules _____ x 74 = D.)_____		
1.2 Calculate the processor input scan of your specialty I/O modules.		
Number of 1746–HS _____ x 312 = E.)_____		
Number of 1/4 DCM _____ x 297 = F.)_____		
Number of 1/2 DCM _____ x 332 = G.)_____		
Number of 3/4 DCM _____ x 366 = H.)_____		
Number of full DCM _____ x 400 = I)_____		
Number of 1747–KE _____ x 423 = J)_____		
Number of 1746–NI4 _____ x 296 = K.)_____		
Number of 1746–NIO4I, NIO4V _____ x 252 = L.)_____		
Number of 1746–NT4 _____ x 365 = M.)_____		
Number of 7–block DSN _____ x 403 = N.)_____		
Number of 30–block DSN _____ x 1031 = O.)_____		
Number of 1746–BAS (SLC 5/01 configuration) _____ x 431 = P.)_____		
1.3 Number of 1747–SN _____ x 1198 = Q.)_____		
Number of 1746–HSCE _____ x 486 = R.)_____		
Number of 1746–BAS (SLC 5/02 & SLC 5/03 configuration) _____ x 585 = S.)_____		
1.4 If S:33/8 is clear, add 78 µs per configured input slot _____ x 78 = T.)_____		
1.5 Add lines A through T. If any values in E through S are > 0, add 6. Place this value on line (W). Add 15 to value on line (W). This is the **minimum input scan.** W.) _____ + 15 = X	**X)**_____	**X)**_____
1.6 Calculate your maximum specialty I/O input scan time: Input scan time (R1) = Number of specialty I/O modules in section 1.2 x 50		**R1)**_____
1.7 Calculate your maximum I/O input scan time for section 1.3: Input scan time (R2) = Number of specialty I/O modules in section 1.3 x 200		**R2)**_____
1.8 Calculate the Forced Input Overhead: Forced Input Overhead = Number of discrete input modules at 3.5 µs per word Add 16.35 + 4.3 per word for each specialty I/O module (i.e. BASIC has 8 inputs = 16.35 + 4.3 x 8 = 50.75) = Y	**Y)**_____	**Y)**_____
1.9.0 Add the values in the minimum scan time columns; put the values in the minimum scan time subtotal.	**subtotal**_____	
1.9.1 Add the values in the maximum scan time columns; put the values in the maximum scan time subtotal.		**subtotal**_____

Continued on next page

Procedure	Min Scan Time	Max Scan Time
2. Estimate your *output* scan time (µs).		
2.1 Calculate the processor output scan of your discrete output modules.		
Number of 8 point modules _____ x 31 = A.)_____		
Number of 16 point modules _____ x 31 = B.)_____		
Number of 32 point modules _____ x 62 = C.)_____		
Number of I/O Combo modules _____ x 82 = D.)_____		
2.2 Calculate the processor output scan of your specialty I/O modules.		
Number of 1746–HS _____ x 349 = E.)_____		
Number of 1/4 DCM _____ x 315 = F.)_____		
Number of 1/2 DCM _____ x 360 = G.)_____		
Number of 3/4 DCM _____ x 404 = H.)_____		
Number of full DCM _____ x 449 = I.)_____		
Number of 1746–NIO4I & NIO4V _____ x 277 = J.)_____		
Number of 1746–NO4I & NO4V _____ x 322 = K.)_____		
Number of 7–block DSN _____ x 449 = L.)_____		
Number of 30–block DSN _____ x 1204 = M.)_____		
Number of 1746–BAS (SLC 5/01 configuration) _____ x 480 = O.)_____		
2.3 Calculate the processor output scan of your specialty I/O modules.		
Number of 1747–SN _____ x 1375 = P.)_____		
Number of 1746–HSCE _____ x 374 = Q.)_____		
Number of 1746–BAS (SLC 5/02 & SLC 5/03 configuration) _____ x 636 = R.)_____		
2.4 If S:33/8 is clear, add 78 µs per configured output slot _____ x 78 = S.)_____		
2.5 Add lines A through S. If any values in E through R are > 0, add 6. Place this value on line (T). Add 12 to value on line (T). This is your **minimum output** scan of your specialty I/O . T.)_____ + 12 =X	**X)**_____	**X)**_____
2.6 Calculate your maximum specialty output scan time: Output scan time (R1) = Number of specialty I/O modules in 2.2 x 50)		**R1)**_____
2.7 Calculate your maximum output scan time for section 2.3: Output scan (R2) = Number of specialty I/O modules x 200		**R2)**_____
2.8 Calculate the Forced Output Overhead: Forced Output Overhead = Number of discrete output modules at 3.0 µs per word Add 16.35 + 4.3 per word for each specialty I/O module (i.e. BASIC has 8 inputs = 16.35 + 4.3 x 8 = 50.75) = Y	**Y)**_____	**Y)**_____
2.9.0 Add the values in the minimum scan time columns; put the values in the minimum scan time subtotal.	**subtotal**_____	
2.9.1 Add the values in the maximum scan time columns; put the values in the maximum scan time subtotal.		**subtotal**_____

Continued on next page

Procedure	Min Scan Time	Max Scan Time
3. Estimate your *program* scan time. This estimate assumes operation of all instructions once per operating scan. 3.1 Count the number of rungs in your ladder program. Multiply this number by 0.187. Place the value on line Rungs. 3.2 Calculate your program execution time when all instructions are true. (See appendix C to do this.) Place this value on line Time. 3.3 Add the values from 3.1 and 3.2; place this on the subtotal line.	**Rungs)**_____ **+** **Time)**_____ **subtotal**_____	**Rungs)**_____ **+** **Time)**_____ **subtotal**_____
4. *Processor* overhead time (S:33/8 = 1 add 665 µs; S:33/8 = 0 add 742 µs)	_____**subtotal**	_____**subtotal**
5. Estimate your *communication* overhead: 5.1 Calculate the background communications overhead using the table below. The minimum overhead applies when devices *are not* connected to a channel.		

Channel 0 (RS232)	Channel 1 (DH+ or Ethernet)	Background Communication Overhead Minimum	Maximum
DF1 Full–Duplex	DH+	1006	1110 (1280)
DF1 Half–Duplex	DH+	1005	1100
DH–485	DH+	1020	1090
DF1 Full–Duplex	Shutdown	1006	1090 (1200)
DF1 Half–Duplex	Shutdown	1005	1080
DH–485	Shutdown	1020	1070

Use the numbers in parentheses when using MSG instructions on DF1 Full-Duplex channel.

5.2 Calculate the foreground communications overhead: for minimum scan time add 0; for maximum scan time add 1027 per channel. (Maximum scan time accounts for programmer being attached to processor.)	_____ Overhead + **0**	_____ Overhead + _____ × **1027**
For DH+ Only –- If S:34/1 is set to enable the active node table update, add 400 per command. If the destination of the packet serviced is the other channel, add 400 per command.	_____**subtotal**	_____**subtotal**
6. Total all the values from the subtotals of steps 1–5 (minimum and maximum input scan, output scan, program scan, processor overhead, and communication overhead).	_____µsecs.	_____µsecs.
7. Convert µsecs. to msecs. by dividing by 1000.	/ 1000	/ 1000
Estimated minimum and maximum scan times for your SLC 5/04 or SLC 5/05 processor application:	msecs.	msecs.

Scan Time for I/O Modules

Use the following maximum scan times to help you calculate your input and output scan times. Refer to worksheet E in this appendix.

SLC 5/03 Processor

I/O Module	Maximum Input Scan Time	Maximum Output Scan Time
1746–8 Point Discrete Input	26	NA
1746–8 Point Discrete Output	NA	31
1746–16 Point Discrete Input	26	NA
1746–16 Point Discrete Output	NA	31
1746–32 Point Discrete Input	52	NA
1746 32 Point Discrete Output	NA	62
1746–IO4, –IO8, –IO12 Discrete I/O Combination	74	82
1746–HS IMC 110 Servo Controller Module	332	369
1747–DCM Direct Communication Module – 1/4 chassis configuration	317	335
1747–DCM Direct Communication Module– 1/2 chassis configuration	352	380
1747–DCM Direct Communication Module– 3/4 chassis configuration	386	424
1747–DCM Direct Communication Module – full chassis configuration	420	469
1746–NI4 Analog Input Module	316	NA
1746–NIO4I, NIO4V Analog Combination Modules	272	297
1746–NO4I, NO4V Analog Output Modules	NA	342
1747–DSN Distributed I/O Scanner Module – 7 block configuration	423	469
1747–DSN Distributed I/O Scanner Module – 30 block configuration	1051	1224
1746–BAS BASIC Module SLC 5/01 configuration	451	500
1746–HSCE High–Speed Counter Encoder Module	506	394
1747–SN Remote I/O Scanner Module	1218	1395
1746–BAS BASIC Module SLC 5/02 and SLC 5/03 configuration	605	656

SLC 5/04 or SLC 5/05 Processor

I/O Module	Maximum Input Scan Time	Maximum Output Scan Time
1746–8 Point Discrete Input	26	NA
1746–8 Point Discrete Output	NA	31
1746–16 Point Discrete Input	26	NA
1746–16 Point Discrete Output	NA	31
1746–32 Point Discrete Input	52	NA
1746 32 Point Discrete Output	NA	62
1746–IO4, –IO8, –IO12 Discrete I/O Combination	74	82
1746–HS IMC 110 Servo Controller Module	312	349
1747–DCM Direct Communication Module – 1/4 chassis configuration	297	315
1747–DCM Direct Communication Module– 1/2 chassis configuration	332	360
1747–DCM Direct Communication Module– 3/4 chassis configuration	366	404
1747–DCM Direct Communication Module – full chassis configuration	400	449
1746–NI4 Analog Input Module	296	NA
1746–NIO4I, NIO4V Analog Combination Modules	252	277
1746–NO4I, NO4V Analog Output Modules	NA	322
1747–DSN Distributed I/O Scanner Module – 7 block configuration	403	449
1747–DSN Distributed I/O Scanner Module – 30 block configuration	1031	1204
1746–BAS BASIC Module SLC 5/01 configuration	431	480
1746–HSCE High–Speed Counter Encoder Module	486	374
1747–SN Remote I/O Scanner Module	1198	1375
1746–BAS BASIC Module SLC 5/02 and SLC 5/03 configuration	585	636

Example Scan Time Calculation

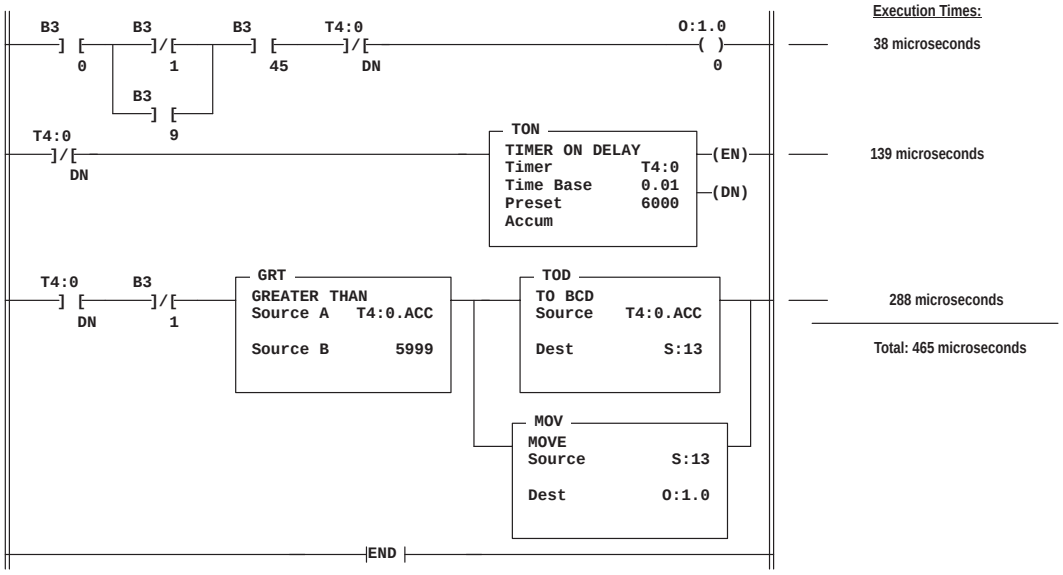
Suppose you have a system consisting of the following components:

System Configuration		Description
Catalog Number	Quantity	
1747–L514	1	4K Processor
1746–IA8	2	8 point 120 VAC Input Module
1746–IB16	1	16 point 24 VDC Sinking Input Module
1746–OA16	3	16 point 120 VAC Relay Output Module
1746–OB8	1	16 point 24 VDC Sourcing Output Module
1746–NIO4V	1	4 Channel Analog Combination Module

Since the 1747-L514 processor is used, worksheet C must be filled out. This is shown on page D–11.

The following ladder program is used in this application. The execution times for the instructions (true state) have been taken from appendix C and added up for each rung. The total execution time, 465 microseconds, is entered in the worksheet on page D–25.

The worksheet indicates that the total estimated scan time is 3.85 milliseconds minimum and 8.9 milliseconds maximum.



Example: Worksheet C – Estimating the Scan Time of an SLC 5/01 Processor Application

Procedure:	Min Scan Time:	Max Scan Time:
1. Estimate your *input* scan time (µs).		
1.1 Calculate the processor input scan of your discrete input modules.		
Number of 8 point modules 2 x 197 = A.) 394		
Number of 16 point modules 1 x 313 = B.) 313		
Number of 32 point modules 0 x 545 = C.) 0		
1.2 Calculate the processor input scan of your specialty I/O modules.		
Number of 1/4 DCM or analog combo 1 x 652 = D.) 652		
Number of 1/2 DCM, analog input, or 1746–HS 0 x 1126 = E.) 0		
Number of 3/4 DCM 0 x 1600 = F.) 0		
Number of full DCM, BASIC, or 1747–DSN 0 x 2076 = G.) 0		
Number of 1747–KE 0 x 443 = H.) 0		
1.3 Add lines A through H. Place this value on line (I). Add 101 to the value on line (I). This sum is your *minimum* input scan time. I.) **1359** + 101 =	**1460**	
1.4 Calculate your *maximum* input scan time: Maximum input scan time = Minimum scan time + (Number of specialty I/O modules x 50)		**1510**
1.5 Calculate the Forced Input Overhead: Forced Input Overhead = (Number of input modules x 180) + 140 per additional word for multi–word modules (e.g. DCM, analog, DSN)		**860**
2. Estimate your *output* scan time (µs).		
2.1 Calculate the processor output scan of your discrete output modules.		
Number of 8 point modules 1 x 173 = A.) 173		
Number of 16 point modules 3 x 272 = B.) 816		
Number of 32 point modules 0 x 470 = C.) 0		
2.2 Calculate the processor output scan of your specialty I/O modules.		
Number of 1/4 DCM or analog combo 1 x 620 = D.) 620		
Number of 1/2 DCM, analog output, or 1746–HS 0 x 1028 = E.) 0		
Number of 3/4 DCM 0 x 1436 = F.) 0		
Number of full DCM, BASIC, or 1747–DSN 0 x 1844 = G.) 0		
2.3 Add lines A through G. Place this value on line (H). Add 129 to the value on line (H). This sum is your *minimum* output scan time. H.) **1609** + 129 =	**1747**	
2.4 Calculate your *maximum* output scan time: Maximum output scan time = Minimum scan time + (Number of specialty I/O modules x 50)		**1788**
2.5 Calculate the Forced Output Overhead: Forced Output Overhead = (Number of output modules x 172) + 140 per additional word for multi–word modules (e.g. DCM, analog, DSN)		**1000**

Continued on next page

Procedure:	Min Scan Time:	Max Scan Time:
3. Estimate your *program* scan time. This estimate assumes operation of all instructions once per operating scan. 3.1 Count the number of rungs in your program. Place value on line (A). 3.2 Calculate your program execution time (B) when all instructions are true. (See appendix C to do this.) **4. Add the values in the minimum and maximum scan time columns.**	**3** **465** **3675** subtotal	**3** **465** **5626** subtotal
5. Add *processor* overhead time (178 for min scan time; 278 for max. scan time) to the subtotals estimated in step 4. Use these new subtotals to calculate communication overhead in step 6.	+ 178 **3853** subtotal	+ 278 **5804** subtotal
6. Estimate your *communication* overhead: 6.1 Calculate the background communication overhead: multiply the subtotal for minimum scan time (estimated in step 5) by 1; multiply the subtotal for maximum scan time by 1.140 (max. value accounts for active DH–485 link). 6.2 Calculate the foreground communication overhead: for minimum scan time add 0; for maximum scan time add 2310. (Maximum scan time accounts for programmer being attached to processor.) 6.3 Convert µsecs. to msecs., divide by 1000.	x 1.000 **3853** µsecs. + 0 **3853** µsecs. / 1000	x 1.140 **6617** µsecs. + 2310 **8927** µsecs. / 1000
Estimated minimum and maximum scan times for your SLC 5/01 processor application:	**3.85** msecs.	**8.9** msecs.

E Programming Instruction References

This appendix lists all of the available programming instructions along with their parameters, valid addressing modes, and file types.

Valid Addressing Modes and File Types

The following addressing modes are available:

Addressing Mode	Example
Direct	N7:0
Indexed Direct	#N7:0
Indirect	N7:[N10:3]
Indexed Indirect	#N7:[N10:3]

The following file types are available:

- O Output
- I Input
- S Status
- B Binary
- T Timer
- C Counter
- R Control
- N Integer
- F Float①②
- A ASCII①②
- ST String①②
- M M0/M1③
- Immediate – indicates that a constant is a valid file type

① Supported only by SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.

② Not supported by MicroLogix 1000 controllers, or Fixed SLC, SLC 5/01, and SLC 5/02 processors.

③ Not supported by MicroLogix 1000 controllers, or Fixed SLC and SLC 5/01 processors.

Understanding the Different Addressing Modes

The following descriptions will help you understand how to structure a specific type of address.

Direct Addressing

The data stored in the specified address is used in the instruction. For example:

N7:0 T4:8.ACC
ST20:5

Indexed Addressing

You may specify an address as being indexed by placing the "#" character in front of the address. When an address of this form is encountered in the program, the processor takes the element number of the address and adds to it the value contained in the Index Register S:24, then uses the result as the actual address. For example:

#N7:10 where S:24 = 15
The actual address used by the instruction is N7:25.

Indirect Addressing

You may specify an address as being indirect by replacing the file number, element number, or sub-element number with a [Xf:e.s] symbol. The word address inside of the bracket is queried for a value. The queried value then becomes the file, element, or sub-element portion of the indirect address. For example:

B3:[N10:2] states that the element address of Bit file 3 is contained in address N10:2. Therefore, if N10:2 contains the value 5, B3:[N10:2] indirectly refers to address B3:5. Other examples include:

N7:[N7:0]	N[N7:0]:[N7:1]
N7:[T4:0.ACC]	C5:[N7:0]

Indexed Indirect Addressing

You may specify a combination of indirect and indexed addressing. The processor first resolves the indirect portion of the address and then adds the offset from the Index Register S:24 to come up with the final address. For example:

#N7:[N10:3] where N10:3 = 20 and S:24 = 15
The actual address used by the instruction is N7:35.

Instruction	Description	Instruction Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
ABL①	ASCII Test Buffer for Line	channel			0
		control	direct	R	none
		characters			0–1024
ABS②	Absolute Value	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
ACB①	ASCII Number of Characters in Buffer	channel			0
		control	direct	R	none
		characters			0–1024
ACI①	ASCII String to Integer	source	direct, indirect	ST	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
ACL①	ASCII Clear Buffer	channel			0
		transmit buffer			0=no or 1=yes
		receive buffer			0=no or 1=yes
ACN①	ASCII String Concatenate	source A	direct, indirect	ST	none
		source B	direct, indirect	ST	none
		destination	direct	ST	none
ACS②	Arc Cosine	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none

① Supported only by SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.

② Supported only by SLC 5/03 (OS302), and SLC 5/04 (OS401), and SLC 5/05 processors.

Instruction	Description	Instruction Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
ADD	Add	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
AEX①	ASCII String Extract	source	direct, indirect	ST	none
		index	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	1–82
		number	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	1–82
		destination	direct	ST	none
AHL①	ASCII Set/Reset Handshake Lines	channel			0
		AND mask	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	0–FFFF
		OR mask	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	0–FFFF
		control	direct	R	none
		channel status			0–001F
AIC①	ASCII Integer to String	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct	ST	none
AND	Logical AND	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none

① Supported only by SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
ARD①	ASCII Read Characters	channel			0
		destination	direct	ST	none
		control	direct	R	none
		string length			0–82
		characters read			0–82
ARL①	ASCII Read Line	channel			0
		destination	direct	ST	none
		control	direct	R	none
		string length			0–82
		characters read			0–82
ASC①	ASCII String Search	source	direct, indirect	ST	none
		index	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	1–82
		search	direct, indirect	ST	none
		result	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
ASN②	Arc Sine	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
ASR①	ASCII String Compare	source A	direct, indirect	ST	none
		source B	direct, indirect	ST	none
ATN②	Arc Tangent	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none

① Supported only by SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
AWA①	ASCII Write with Append	channel			0
		source	direct	ST	none
		control	direct	R	none
		string length			0–82
		characters sent			0–82
AWT①	ASCII Write	channel			0
		source	direct	ST	none
		control	direct	R	none
		string length			0–82
		characters sent			0–82
BSL	Bit Shift Left	file	indexed direct indexed indirect	O, I, S, B, N, A, ST	none
		control	direct	R	none
		bit address	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		length			0–2048
BSR	Bit Shift Right	file	indexed direct indexed indirect	O, I, S, B, N, A, ST	none
		control	direct	R	none
		bit address	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		length			0–2048
CLR	Clear	destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none

① Supported only by SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
COP	Copy File	source	indexed direct indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		destination	indexed direct indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		length			1–128
COS②	Cosine	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
CPT②	Compute	destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		expression			<expression>
CTD	Count Down	counter	direct	C	none
		preset			–32,768–32,767
		accum			–32,768–32,767
CTU	Count Up	counter	direct	C	none
		preset			–32,768–32,767
		accum			–32,768–32,767
DCD	Decode 4 to 1 of 16	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
DDV	Double Divide	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
DEG②	Radians to Degrees	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
DIV	Divide	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
ENC③	Encode 1 of 16 to 4	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
EQU	Equal	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
FFL④	FIFO Load	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M⑦	–32,768–32,767
		FIFO array	indexed direct indexed indirect	O, I, S, B, N, A	none
		FIFO control	direct	R	none
		length			1–128
		position			0–127
FFU④	FIFO Unload	FIFO array	indexed direct indexed indirect	O, I, S, B, N, A	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M⑦	none
		FIFO control	direct	R	none
		length			1–128
		position			0–127

③ Supported only by MicroLogix 1000 controllers.

④ Not supported by SLC 5/01 processors and Fixed controllers.

⑦ Indexed addressing is not allowed when using T, C, R, or M addresses.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
FLL	Fill File	source	direct, indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	indexed direct indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		length			1–128
FRD	From BCD to Binary	source (SLC 5/01)	direct	O, I, S, B, T, C, R, N, A, ST, M	none
		source (SLC 5/02, 5/03, 5/04, 5/05, and MicroLogix 1000)	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
GEQ	Greater Than or Equal	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
GRT	Greater Than	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
HSC⑥	High–Speed Counter (SLC 5/01)	counter			none
		preset			1–32,767
HSC③	High–Speed Counter	type			0–7
		counter	direct	C	none
		preset			–32,768–32,767
		accum			–32,768–32,767

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
HSD③	HSC Interrupt Disable	counter	direct	C	none
HSE③	HSC Interrupt Enable	counter	direct	C	none

③ Supported only by MicroLogix 1000 controllers.

⑥ Supported only by L20, L30, and L40 Fixed SLC processors with DC inputs.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
HSL[3]	HSC Load	counter	direct	C	none
		source	direct	B and N	none
		length			always 5
IID[7]	I/O Interrupt Disable	slots			double hex word (list of slots)
IIE[7]	I/O Interrupt Enable	slots			double hex word (list of slots)
IIM	Immediate Input with Mask	slot	direct	I	none
		mask	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		length (SLC 5/03, 5/04, and 5/05)			1–32
INT[4]	I/O Interrupt				none
IOM	Immediate Output with Mask	slot	direct	O	none
		mask	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		length (SLC 5/03, 5/04, and 5/05)			1–32
JMP	Jump	label number			0–999
JSR	Jump to Subroutine	subroutine file number			3–255
LBL	Label Declaration	label number			0–999

[3] Supported only by MicroLogix 1000 controllers.

[4] Not supported only by SLC 5/01 processors and Fixed controllers.

[7] Supported only by SLC 5/02, SLC 5/03, SLC 5/04, and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
LEQ	Less Than or Equal To	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
LES	Less Than	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
LFL④	LIFO Load	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M⑤	–32,768–32,767
		LIFO array	indexed direct indexed indirect	O, I, S, B, N, A	none
		LIFO control	direct	R	none
		length			1–128
		position			0–127
LFU④	LIFO Unload	LIFO array	indexed direct indexed indirect	O, I, S, B, N, A	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M⑤	none
		LIFO control	direct	R	none
		length			1–128
		position			0–127
LIM④	Limit Test (circ)	low limit	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		test	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		high limit	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max

④ Not supported only by SLC 5/01 processors and Fixed controllers.

⑤ Indexed addressing is not allowed when using T, C, R, or M addresses.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
LN②	Natural Log	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
LOG②	Log to the Base 10	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
MCR	Master Control Relay				none
MEQ	Mask Compare Equal To	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		source mask	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		compare	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
MOV	Move	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
MSG (5/02 only)	Message	read/write			0=read,1=write
		target device			2=500CPU, 4=485CIF
		control block	direct	N	none
		control block length			7
		local address	direct	O, I, S, B, T, C, R, N, A	none
		target node			0–31
		target address	direct	O, I, S, B, T, C, R, N, A	0–255

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
		message length		T, C, R	1–13
				I, O, S, B, N	1–41

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
MSG (5/03, 5/04, and 5/05[9])	Message	type			64=peer-to-peer
		read/write			0=read, 1=write
		target device			2=500CPU, 4=485CIF, 8=PLC5
		local/remote			16=local, 32=remote
		control block	direct	N	none
		control block length			14[10]
		channel number			5/03 and 5/04: 0 or 1 5/05: 0 only
		target node			0–31, 0–254 if 485CIF
		remote bridge link ID			0–254, 0 when local
		remote bridge node address			0–254, 0 when local
		local bridge node address			0–254, 0xFFFF when local
		local file address	direct	O, I, S, B, T, C, R, N, F, A, ST, M[8]	none
		target file address	direct	O, I, S, B, T, C, R, N, F, A, ST, M[8]	0–255
		message length		O, I, B, N, A[8]	1–103
				S[8]	5/03 and 5/05: 1–83 5/04: 1–164
				F[8]	1–51
				T	1–34 (if PLC5: 1–20)
				C, R	1–34
				ST[8]	2 (if PLC5: 1)
		message timeout			0–255 (SLC 5/05–23 seconds, read only)

[8] File types F, A, and ST only apply to SLC 5/03 (OS301 or later), SLC 5/04, and SLC 5/05 processors.

[9] SLC 5/05 Channel 0 (RS-232 serial port) only.

[10] For SLC 5/05, control block length = 55 if logical ASCII addressing is used.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
MSG (5/05 Ethernet)	Message	type			64=peer-to-peer
		read/write			0=read, 1=write
		target device			2=500CPU, 4=485CIF, 8=PLC5
		local			16=local
		control block	direct	N	none
		control block length			51 (93 if logical ASCII addressing is used)
		channel number			1 (Ethernet)
		target node, remote bridge link ID, local and remote bridge node address	not applicable		
		IP address (ww.xx.yy.zz)			any legal IP address
		local file address	direct	O, I, S, B, T, C, R, N, F, A, ST, M	none
		target file address	direct	O, I, S, B, T, C, R, N, F, A, ST, M	0–255
		message timeout			0–255

Note *Message lengths for SLC 5/05 processors are shown in the next table.*

Message lengths are based on Ethernet buffer size of 2108 bytes (includes command header and sytem addressing in addition to actual file data). The local file is the destination file for reads and the source files for writes.

In the following table, byte values are the maximum byte length argument values the compiler should generate to be passed to the MSG instruction and then written to word 11 of the MSG control block by the processor at run time.

File Type	485CIF and 500CPU Read/Write		PLC5 Read		PLC5 Write	
	Elements	Bytes	Elements	Bytes	Elements	Bytes
O,I	256	512	256	512	256	512
S	83	166	83	166	83	166
B	256	512	256	512	256	512
T	256	1536	256 (208 when target file type is Timer)	1536 (1248 when target file type is Timer)	208 (201 when using logical ASCII addressing)	1248 (1206 when using logical ASCII addressing)
C	256	1536	256	1536	256	1536
R	256	1536	256	1536	256	1536
N	256	512	256	512	256	512
F	256	1024	256	1024	256	1024
A	256	512 – Treat an ASCII file element here as a whole 16-bit word.	256	512 – Treat an ASCII file element here as 1 byte like PLCs do; compensate by doubling the number of elements that can be read.	256	512 – Treat an ASCII file element here as 1 byte like PLCs do; compensate by doubling the number of elements that can be read.
ST	25 (24 for 500CPU write)	2100 (2016 for 500CPU write)	1		1	
M	256	512	256	512	256	512

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
MUL	Multiply	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
MVM	Masked Move	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		source mask	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
NEG	Negate	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
NEQ	Not Equal To	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
NOT	Logical NOT	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
OR	Logical OR	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
OSR	One–Shot Rising	bit address	direct, indirect	O, I, S, B, T, C, R, N, A, ST	none

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
OTE	Output Energize	bit address	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	none
OTL	Output Latch	bit address	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	none
OTU	Output Unlatch	bit address	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	none
PID⑦	PID	control block	direct	N	none
		process variable	direct, indirect	O, I, B, T, C, R, N, A	none
		control variable	direct, indirect	O, I, B, T, C, R, N, A	none
		control block length			23 always
RAC③	HSC Reset Accumulator	counter	direct	C	none
		source	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767 f-min–f-max
RAD②	Degrees to Radians	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
REF⑦	I/O Refresh	channel 0			0=no, 1=yes
		channel 1			0=no, 1=yes
RES	Timer/Counter Reset	structure	direct	T, C, R	none
RET	Return				none
RPI⑦	Reset Pending Interrupt	slots			double hex word (list of slots)

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

③ Supported only by MicroLogix 1000 controllers.

⑦ Supported only by SLC 5/02, SLC 5/03, SLC 5/04, and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
RTO	Retentive Timer On	timer	direct	T	none
		time base (SLC 5/01)			0.01 only
		time base (SLC 5/02, 5/03, 5/04, 5/05, and MicroLogix 1000)			0.01 or 1.00
		preset			0–32,767
		accum			0–32,767
SBR	Subroutine				none
SCL④	Scale	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		rate	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		offset	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
SCP②	Scale with Parameters	input	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
		input min.	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		input max.	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		scaled min.	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		scaled max.	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		scaled output	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

④ Not supported only by SLC 5/01 processors and Fixed controllers.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
SIN②	Sine	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
SQC	Sequencer Compare	file	indexed direct indexed indirect	O, I, S, B, N, A, ST	none
		mask	direct, indexed direct⑤ indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		source	direct, indexed direct⑤ indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		control	direct	R	none
		length			1–255
		position			0–255
SQL④	Sequencer Load	file	indexed direct indexed indirect	O, I, S, B, N, A, ST	none
		source	direct, indexed direct⑤ indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		control	direct	R	none
		length			1–255
		position			0–255

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

④ Not supported only by SLC 5/01 processors and Fixed controllers.

⑤ Indexed addressing is not allowed when using T, C, R, or M addresses.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
SQO	Sequencer Output	file	indexed direct indexed indirect	O, I, S, B, N, A, ST	none
		mask	direct, indexed direct⑤ indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct, indexed direct⑤ indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
		control	direct	R	none
		length			1–255
		position			0–255
SQR④	Square Root	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
STD④	Selectable Timed Interrupt Disable				none
STE④	Selectable Timed Interrupt Enable				none
STS④	Selectable Timed Interrupt Start	file	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	0, 3–255 except MicroLogix 1000 controllers always equal 5
		time	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	0–255 (SLC 5/02 and MicroLogix 1000), 0–32,767 (SLC 5/03, 5/04, 5/05)
SUB	Subtract	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,76 f-min–f-max
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none

④ Not supported only by SLC 5/01 processors and Fixed controllers.

⑤ Indexed addressing is not allowed when using T, C, R, or M addresses.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
SUS	Suspend	suspend ID			–32,768–32,767
SVC⑦	Service Communications	channel 0 (SLC 5/03, 5/04, 5/05)			0=no, 1=yes
		channel 1 (SLC 5/03, 5/04, 5/05)			0=no, 1=yes
SWP②	Swap	source	indexed direct indexed indirect	B, N, A, ST	none
		length			1–128: bit, 1–128: integer, 1–41: string, 1–128: ASCII
TAN②	Tangent	source	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none
TND	Temporary End				none
TOD	Convert to BCD	source (SLC 5/01)	direct	O, I, S, B, T, C, R, N	none
		source (SLC 5/02, 5/03, 5/04, 5/05)	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	
		destination	direct	O, I. S. B. T, C, R, N, A, ST, M	none
TOF	Timer Off Delay	timer	direct	T	none
		time base (SLC 5/01)			0.01 only
		time base (SLC 5/02, 5/03, 5/04, 5/05, and MicroLogix 1000)			0.01 or 1.00
		preset			0–32,767
		accum			0–32,767

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

⑦ Supported only by SLC 5/02, SLC 5/03, SLC 5/04, and SLC 5/05 processors.

Instruction	Description	Parameter	Valid Addressing Mode(s)	Valid File Types	Immediate Values
TON	Timer On Delay	timer	direct	T	none
		time base (SLC 5/01)			0.01 only
		time base (SLC 5/02, 5/03, 5/04, 5/05, and MicroLogix 1000)			0.01 or 1.00
		preset			0–32,767
		accum			0–32,767
XIC	Examine On (Examine if Closed Contact)	source bit	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	none
XIO	Examine Off (Examine if Open Contact)	source bit	direct, indirect	O, I, S, B, T, C, R, N, A, ST, M	none
XOR	Logical Exclusive OR	address A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		address B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	–32,768–32,767
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, A, ST, M	none
XPY②	X to the Power of Y	source A	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		source B	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	–32,768–32,767 f-min–f-max
		destination	direct, indexed direct indirect, indexed indirect	O, I, S, B, T, C, R, N, F, A, ST, M	none

② Supported only by SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors.

F Data File Organization and Addressing

This chapter discusses the following topics:

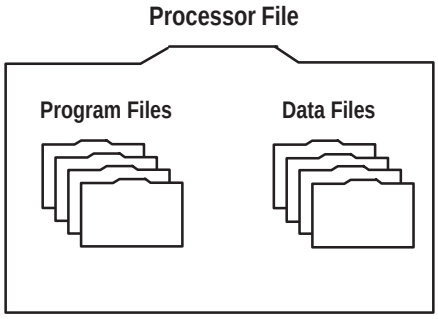
- data file organization and addressing
- specifying indexed addressing
- specifying indirect addressing
  (SLC 5/03 OS302, SLC 5/04 OS401, and SLC 5/05 processors)
- specifying indirect indexed addressing
  (SLC 5/03 OS302, SLC 5/04 OS401, and SLC 5/05 processors)
- addressing file instructions (using the file indicator #)
- numeric constants
- M0-M1 files, G files
  (SLC 5/02 and higher processors with specialty I/O modules)

Understanding File Organization

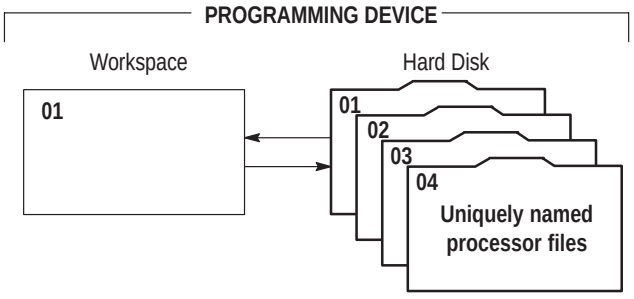
The processor provides control through the use of a program you create, called a processor file. This file contains other files that break your program down into more manageable parts.

Processor File Overview

Most of the operations you perform with the programming device involve the processor file and the two components created with it: program files and data files.



The programming device stores processor files on *hard disk (or floppy disk)*. Monitoring and editing of processor files is done in the *workspace* of the computer. After you select a file from disk and edit it, you then *save* the file hard to disk, replacing the original disk version with the edited version. The hard disk is the recommended location for a processor file.



Processor files are created in the offline mode using the programming device. These files are then restored (downloaded), to the processor for online operation.

Program Files

Program files contain controller information, the main ladder program, interrupt subroutines, and any subroutine programs. These files are:

- **System Program** (file 0) – This file contains various system related information and user-programmed information such as processor type, I/O configuration, processor file name, and password.
- **Reserved** (file 1) – This file is reserved.
- **Main Ladder Program** (file 2) – This file contains user-programmed instructions defining how the controller is to operate.

- **Subroutine Ladder Program** (file 3-255) – These files are user-created and accessed according to subroutine instructions residing in the main ladder program file.

The following program files are specific to the MicroLogix 1000 controllers:

- **User Error Fault Routine** (file 3) – This file is executed when a recoverable fault occurs.
- **High-Speed Counter Interrupt** (file 4) – This file is executed when an HSC interrupt occurs. It can also be used for a subroutine ladder program.
- **Selectable Timed Interrupt** (file 5) – This file is executed when an STI occurs. It can also be used for a subroutine ladder program.
- **Subroutine Ladder Program** (files 6 – 15) – These are used according to subroutine instructions residing in the main ladder program file or other subroutine files.

Note *The MicroLogix 1000 is set up to always have 15 program files. None of these files can be deleted in a MicroLogix 1000 ladder program.*

Data Files

Data files contain the status information associated with external I/O and all other instructions you use in your main and subroutine ladder program files. In addition, these files store information concerning processor operation. You can also use the files to store "recipes" and look-up tables if needed.

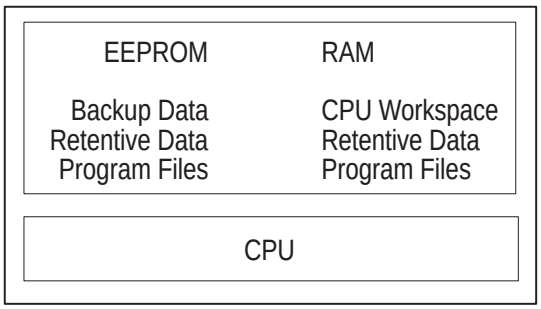
These files are organized by the type of data they contain. The data file types are:

- **Output** (file 0) – This file stores the state of the output terminals for the controller.
- **Input** (file 1) – This file stores the status of the input terminals for the controller.
- **Status** (file 2) – This file stores controller operation information. This file is useful for troubleshooting controller and program operation.
- **Bit** (file 3) – This file is used for internal relay logic storage.
- **Timer** (file 4) – This file stores the timer accumulator and preset values and status bits.
- **Counter** (file 5) – This file stores the counter accumulator and preset values and the status bits.
- **Control** (file 6) – This file stores the length, pointer position, and status bits for specific instructions such as shift registers and sequencers.

- **Integer** (file 7) – This file is used to store numeric values or bit information.
- **Floating Point** (file 8) – This file stores single precision non-extended 32-bit numbers. Applies to SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.
- **String** (user-defined file) – Applies to SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.
- **ASCII** (user-defined file) – Applies to SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.

Understanding How Processor Files are Stored and Accessed

The MicroLogix 1000 programmable controller uses two devices for storing processor files: RAM and EEPROM. The RAM provides easy access storage (i.e., its data is lost on a power down), while the EEPROM provides long-term storage (i.e., its data is not lost on a power down). The diagram below shows how the memory is allocated in the micro controller's processor.

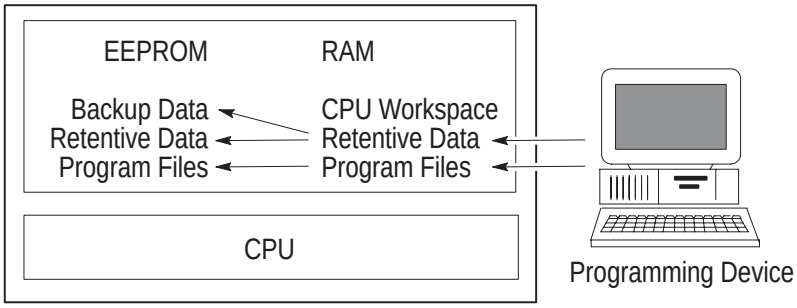


The memory device that is used depends on the operation being performed. This section describes how memory is stored and accessed during the following operations:

- download
- normal operation
- power down
- power up

Download

When the processor file is downloaded to the micro controller, it is first stored in the volatile RAM. It is then transferred to the non-volatile EEPROM, where it is stored as both backup data and retentive data.

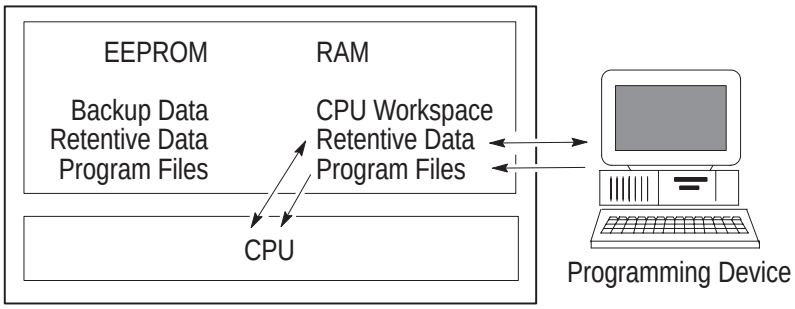


Note *If you want to ensure that the backup data is the same for every micro controller you are using, save the program to disk before downloading it to a micro controller.*

Normal Operation

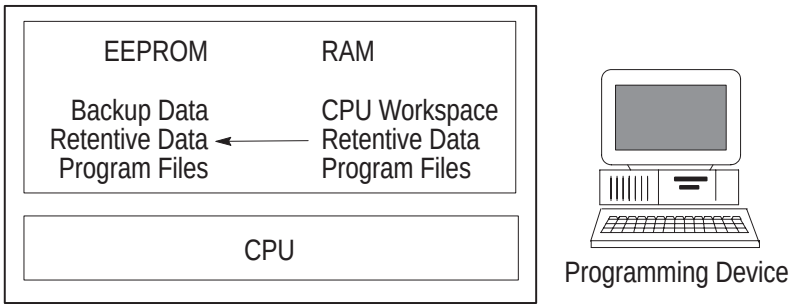
During normal operation, both the micro controller and your programming device can access the processor files stored in the RAM. Any changes to retentive data that occur due to program execution or programming commands affect only the retentive data in the RAM.

The program files are never modified during normal operation. However, both the CPU and your programming device can read the program files stored in RAM.



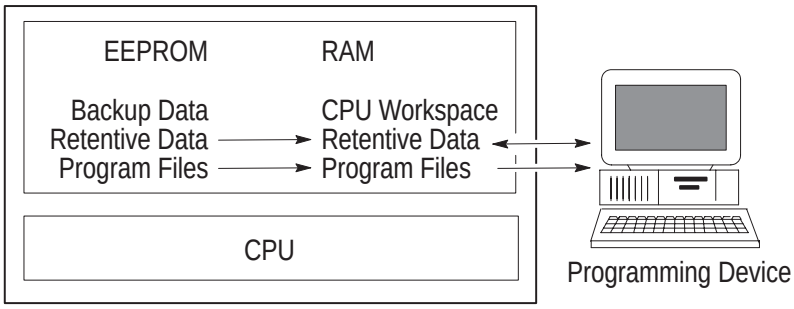
Power Down

When a power down occurs, only the retentive data is transferred from the RAM to the EEPROM. (The program files do not need to be saved to the EEPROM since they cannot be modified during normal operation.) If for some reason power is lost before all of the retentive data is saved to the EEPROM, the retentive data is lost. This may occur due to an unexpected reset or a hardware problem.

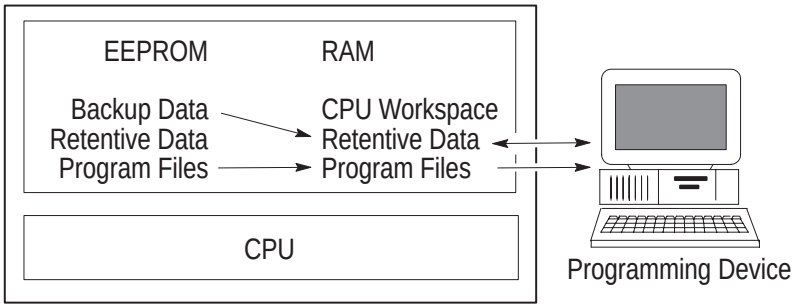


Power Up

During power up, the micro controller transfers the program files from the EEPROM to the RAM. The retentive data is also transferred to the RAM, provided it was not lost on power down, and normal operation begins.



If retentive data was lost on power down, the backup data from the EEPROM is transferred to the RAM and used as the retentive data. In addition, status file bit S2:5/8 (retentive data lost) is set and a recoverable major error occurs when going to run.



Addressing Data Files

For the purposes of addressing, each data file type is identified by a letter (identifier) and a file number.

File numbers 0 through 7 are the default files that fixed, SLC 5/01, SLC 5/02, and SLC 5/03 OS300 processors and MicroLogix 1000 controllers create for you. File number 8 applies only to SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors. If you need additional storage, you can create files by specifying the appropriate identifier and a file number from 9 to 255. Refer to the tables below:

Data file types, identifiers, and numbers (data files in processor memory)

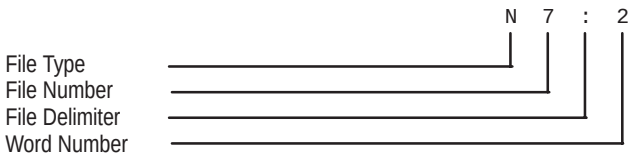
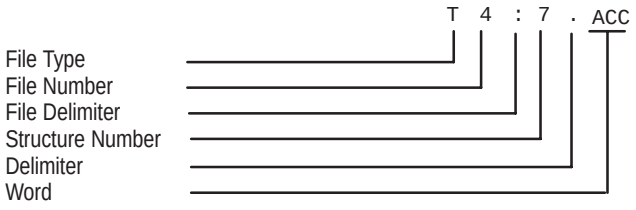
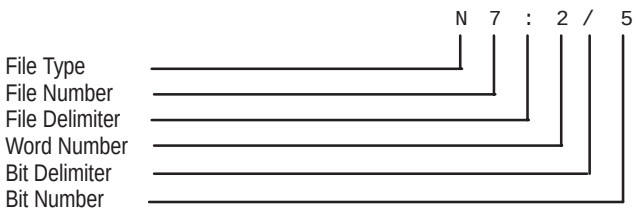
File Type	Identifier	File Number
Output	O	0
Input	I	1
Status	S	2
Bit	B	3
Timer	T	4
Counter	C	5
Control	R	6
Integer	N	7
Float	F	8

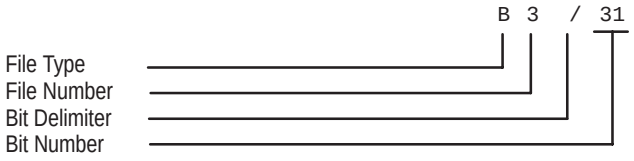
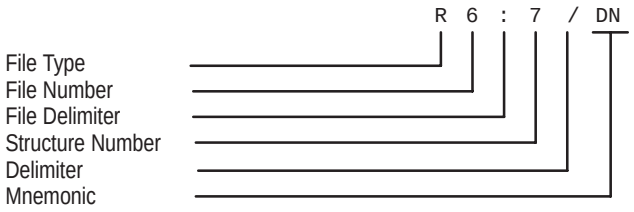
User–Defined Files		
File Type	Identifier	File Number
Bit	B	9–255
Timer	T	
Counter	C	
Control	R	
Integer	N	
Float	F	
String	St	
ASCII	A	

Note *Floating point, string, and ASCII file types are only available when using SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors.*

Specifying Logical Addresses

You assign logical addresses to instructions from the highest level (element) to the lowest level (bit). Addressing examples are shown in the table below.

To specify the address of a:	Use these parameters:①
Word within an integer file	
Word within a structure file (e.g., a timer file)	
Bit within an integer file	

To specify the address of a:	Use these parameters:①
Bit within a bit file	 Bit files are bit stream continuous files, and therefore you can address them in two ways: by word and bit, or by bit alone.
Bit within a structure file (e.g., a control file)	

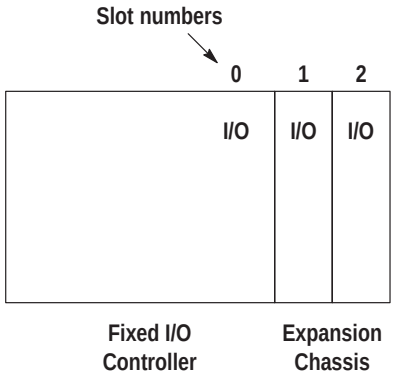
① Some programming devices support short addressing. This allows you to eliminate the file number and file delimiter from addresses. Consult your programming device's user manual for information on addressing capabilities. (For example: N7:2 = N2; T4:12.ACC = T12.ACC; B3:2/12 = B2/12)

You can also address at the bit level using mnemonics for timer, counter, or control data types. The available mnemonics depend on the type of data.

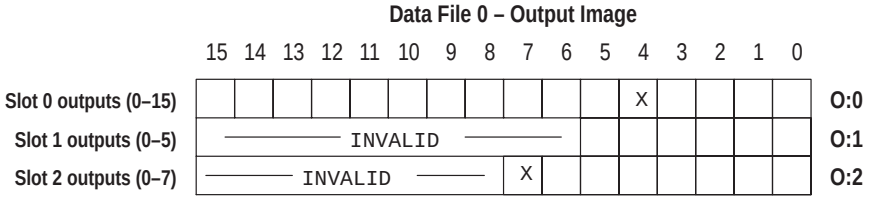
I/O Addressing for a Fixed I/O Controller

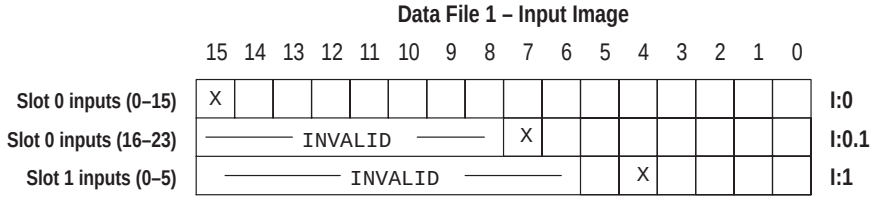
In the following figure, a fixed I/O controller has 24 inputs and 16 outputs. An expansion chassis has been added. Slot 1 of the chassis contains a module having 6 inputs and 6 outputs. Slot 2 contains a module having 8 outputs.

The following figure shows how these outputs and inputs are arranged in data files 0 and 1. For these files, the element size is always 1 word.



Slot	Inputs	Outputs
0	24	16
1	6	6
2	None	8





X = See the examples on the next page.

The table on the following page explains the addressing format for outputs and inputs. Note that the format specifies **e** as the slot number and **s** as the word number. When you are dealing with file instructions, refer to the element as **e . s** (slot and word), taken together.

Assign I/O addresses to fixed I/O controllers as shown in the following table:

Format	Explanation		
O:e.s/b I:e.s/b	O	Output	
	I	Input	
	:	Element delimiter	
	e	Slot number (decimal)	Fixed I/O controller: 0
			Left slot of expansion chassis: 1 Right slot of expansion chassis: 2
	.	Word delimiter. Required only if a word number is necessary as noted below.	
	s	Word number	Required if the number of inputs or outputs exceeds 16 for the slot. Range: 0- 255 (range accommodates multi-word “specialty cards”)
	/	Bit delimiter	
	b	Terminal number	Inputs: 0- 15 (or 0 to 23, slot 0) Outputs: 0- 15

Examples (applicable to the controller shown on page F–10):

O:0/4	Controller output 4 (slot 0)
O:2/7	Output 7, slot 2 of the expansion chassis
I:1/4	Input 4, slot 1 of the expansion chassis
I:0/15	Controller input 15 (slot 0)
I:0.1/7	Controller input 23 (bit 07, word 1 of slot 0)

Alternate way of addressing I/O terminals 16 and higher: As indicated above, address I:0.1/7 applies to input terminal 23 of slot 0. You can also address this terminal as I:0/23.

Word addresses:

O:1	Output word 0, slot 1
I:0	Input word 0, slot 0
I:0.1	Input word 1, slot 0

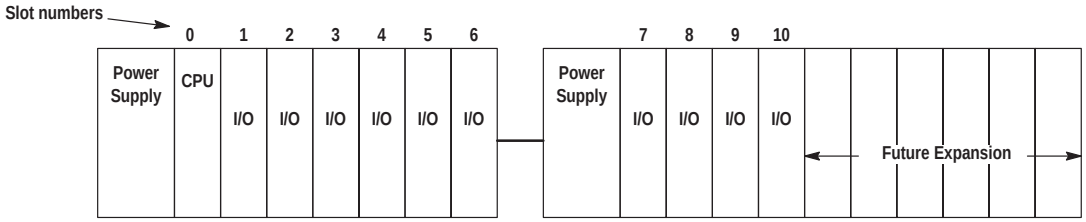
Default Values: Your programming device will display an address more formally. For example, when you assign the address I:1/4, the programming device will show it as I:1.0/4 (Input file, slot 1, word 0, terminal 4).

I/O Addressing for a Modular Controller

With modular controllers, slot number 0 is reserved for the processor module (CPU). Slot 0 is invalid as an I/O slot.

The figure below shows a modular controller configuration consisting of a 7-slot chassis interconnected with a 10-slot chassis. Slot 0 contains the CPU. Slots 1 through 10 contain I/O modules. The remaining slots are saved for future I/O expansion.

The figure indicates the number of inputs and outputs in each slot and also shows how these inputs and outputs are arranged in the data files. For these files, the element size is always 1 word.



Modular controller using a 7-slot chassis interconnected with a 10-slot chassis.

Slot	Inputs	Outputs
1	6	6
2	32	None
3	None	16
4	8	8
5	None	32
6	16	None
7	16	None
8	8	None
9	None	16
10	None	16

Data File 0 – Output Image

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Slot 1 outputs (0–5)	INVALID																O:1
Slot 3 outputs (0–15)	X																O:3
Slot 4 outputs (0–7)	INVALID																O:4
Slot 5, word 0 outputs (0–15)																X	O:5
Slot 5, word 1 outputs (0–15)																	O:5.1
Slot 9 outputs (0–15)																	O:9
Slot 10 outputs (0–15)					X												O:10

Data File 1 – Input Image

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
Slot 1 inputs (0–5)	INVALID																I:1
Slot 2, word 0 inputs (0–15)																	I:2
Slot 2, word 1 inputs (0–15)													X				I:2.1
Slot 4 inputs (0–7)	INVALID																I:4
Slot 6 inputs (0–15)																	I:6
Slot 7 inputs (0–15)								X									I:7
Slot 8 inputs (0–7)	INVALID																I:8

x = See the examples on the next page.

Specifying Indexed Addresses

The indexed address symbol is the # character. Place the # character immediately before the file-type identifier in a logical address. You can use more than one indexed address in your ladder program.

Enter the offset value in word 24 of the status file (S:24). All indexed instructions use the same word S:24 to store the offset value. The processor starts operation at the base address plus the offset. You can manipulate the offset value in your ladder logic before each indexed address operation.

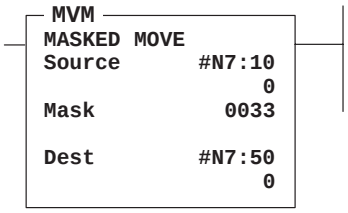
When you specify indexed addresses, follow these guidelines:

- Make sure the index value (positive or negative) does not cause the indexed address to exceed the file type boundary.
- When an instruction uses more than two indexed addresses, the processor uses the same index value for each indexed address.
- Set the index word to the offset value you want immediately before enabling an instruction that uses an indexed address.

Instructions with a # sign in an address manipulate the offset value stored at S:24. Make sure you monitor or load the offset value you want prior to using an indexed address. Otherwise unpredictable machine operation could occur with possible damage to equipment and/or injury to personnel.

Example of Indexed Addressing

The following Masked Move (MVM) example uses an indexed address in the source and destination addresses. If the offset value is 10 (stored in S:24), the processor manipulates the data stored at the base address plus the offset.



In this example, the processor uses the following addresses:

Value:	Base Address:	Offset Value in S:24	Offset Address:
Source	N7:10	10	N7:20
Destination	N7:50	10	N7:60

SLC 5/03 (OS301 and higher), SLC 5/04, and SLC 5/05 processors — If the indexed address is a floating point (F8:) data file, then the index offset value in S:24 is the offset in elements. If the indexed address is a string (ST) data file, then the index offset value in S:24 is the offset in sub-elements. This limits string element boundaries from being crossed.

Note that file instructions (SQO, COP, LFL for example) overwrite S:24 when they execute. For this reason, you must insure that the index register is loaded with the intended value prior to the execution of an indexed instruction that follows a file instruction.

Creating Data for Indexed Addresses

Data tables are not expanded automatically to accommodate indexed addresses. You must create this data with the memory map function. In the example on the previous page, data words N7:3 through N7:12 and N11:6 through N11:15 must be allocated. Failure to do so will result in an unintended overwrite condition or a major fault.

Crossing File Boundaries

An offset value may extend operation to an address outside the data file boundary. You can either allow or disallow crossing file boundaries. If you choose to disallow crossing file boundaries, a runtime error occurs if you use an offset value which would result in crossing a file boundary.

SLC 5/02 specific — You are allowed to select crossing file boundaries only if no indexed addresses exist in the O: (output), I: (input), or S: (status) files. This selection is made at the time you save your program. The file order from start to finish is:

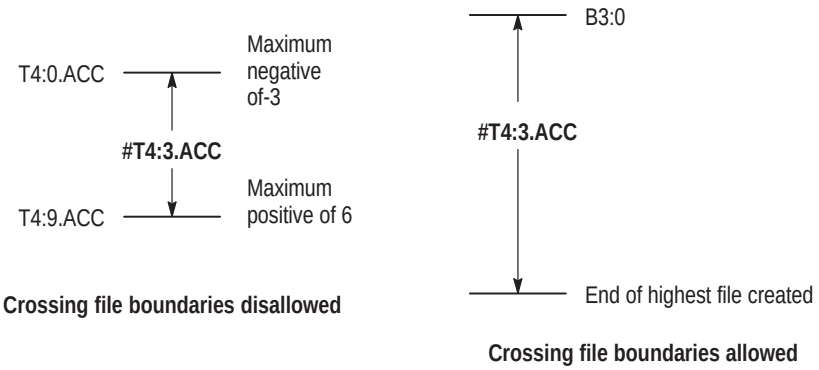
- **O0:, I1:, S2:, B3:, T4:, C5:, R6:, N7:, x9:, x10: . . .**
- x9: and x10: . . . are application-specific files where x can be of types B, T, C, R, N.

SLC 5/03 (OS301 and higher) SLC 5/04, and SLC 5/05 processors- When an indexed string data file is specified, indexed addressing is not allowed to cross a string element boundary. A run-time error will occur if you use an offset value that results in crossing a string element boundary.

Note *If a file is constant protected, indexing across file boundaries is not allowed.*

Example

The figure below indicates the maximum offset for word address #T4:3.ACC when allowing and disallowing crossing file boundaries.



Crossing file boundaries disallowed: In the example above, the highest numbered element in the timer data file is T4:9. This means that #T4:3.ACC can have a maximum negative offset of-3 and a maximum positive offset of 6.

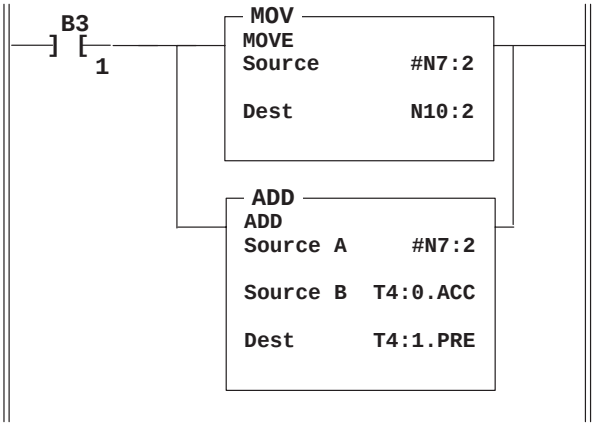
Crossing file boundaries allowed: The maximum negative offset extends to the beginning of data file 3. The maximum positive offset extends to the end of the highest numbered file created.

Monitoring Indexed Addresses

The offset address value is not displayed when you monitor an indexed address. For example, the value at N7:2 appears when you monitor indexed address #N7:2.

Example

If your application requires you to monitor indexed data, we recommend that you use a MOV instruction to store the value.



N10:2 will contain the data value that was added to T4:0.ACC.

File Instructions

The # symbol is also required for addresses in file instructions. The indexed addresses used in these file instructions also make use of word S:24 to store an offset value upon file instruction completion. Refer to the next page for a list of file instructions that use the # symbol for addressing.

File instructions manipulate the offset value stored in word S:24. Make sure that you load the correct offset value in S:24 prior to using an indexed address that follows a file instruction. Otherwise, unpredictable operation could occur, resulting in possible personal injury and/or damage to equipment.

Effects of Program Interrupts on Index Register S:24

When normal program operation is interrupted by the user error handler, an STI, or an I/O interrupt, the content of index register S:24 is saved; then, when normal program operation is resumed, the content of index register S:24 is restored. This means that if you alter the value in S:24 in these interrupt subroutines, the system will overwrite your alteration with the original value contained on subroutine entry.

Specifying an Indirect Address

Indirect addressing allows you to write less complex ladder logic programs and saves you memory space. You have the option of using word-level and bit-level indirect addresses when using an SLC 5/03 (OS302), SLC 5/04 (OS401), and SLC 5/05 processors. Indirect bit addresses are based on the form of the indirect address and the type of bit instruction.

Use indirect addressing for applications such as indexing sequential batch files in a multiple batch operation. For example, at completion of each operation, let a counter accumulated value call out the next batch file, such as:
N10, N11, N12,...N[C5:0.ACC].

When you specify indirect addresses, follow these guidelines:

- You can indirectly address:
  - file number
  - word number (element + subelement)
  - bit number (in a binary file)
- The substitute address must be any address specified to the word level.
- Enter the substitute address in brackets [ ].

Examples

Valid Address	Variable	Explanation
N7:[C5:7.ACC]	Word number	The word number is the accumulated value of counter 7 in file 5.
B3/[I:0.17]	Bit number	The bit number is stored in input word 17.
N[N7:0]:[N9:1]	File and word number	The file number is stored in integer address N7:0 and the word number in integer address N9:1.
St10:[N7:0].1	Element number	The element number is stored in N7:0.
I:[N7:0].1/1	Slot number	The slot number is stored in N7:0.

Creating Data for Indirect Addresses

Data tables are not expanded automatically to accommodate indirect addresses. You must create this data with your programming software.

Crossing File Boundaries

Crossing file boundaries is not allowed. A runtime error occurs if you use an offset value which would result in crossing a file boundary.

Monitoring Indirect Addresses

An asterisk is displayed at all times when monitoring an indirect address.

Addressing File Instructions – Using the File Indicator (#)

File instructions employ user-created files. These files are addressed with the # sign. They store an offset value in word S:24, just as with indexed addressing discussed in the last section.

COP	Copy File	**LFL**	(LIFO Load)*
FLL	Fill File	**LFU**	(LIFO Unload)*
BSL	Bit Shift Left	**SQO**	Sequencer Output
BSR	Bit Shift Right	**SQC**	Sequencer Compare
FFL	(FIFO Load)*	**SQL**	Sequencer Load*
FFU	(FIFO Unload)*		

* Available in the SLC 5/02 and higher processors.

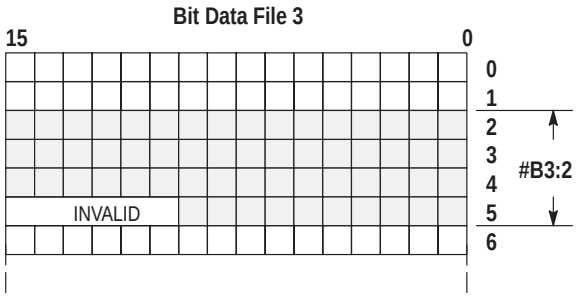
If you are using file instructions and also indexed addressing, make sure that you monitor and/or load the correct offset value prior to using an indexed address. Otherwise, unpredictable operation could occur, resulting in possible personal injury and/or damage to equipment.

The following paragraphs explain user-created files as they apply to Bit Shift instructions, Sequencer instructions, Copy File, and Fill File instructions.

Bit Shift Instructions

The following figure shows a user-defined file within bit data file 3. For this particular user-defined file, you would enter the following parameters when programming the instruction:

Address of the bit array is **#B3:2**
Length of the bit array is 58, entered as a separate parameter in the Bit Shift instruction.

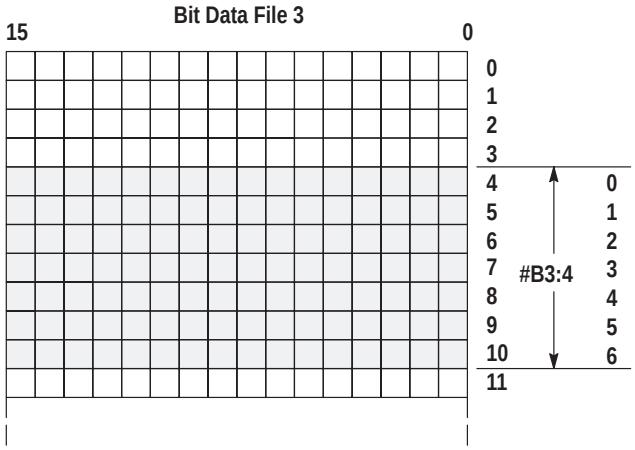


- **#B3:2** The address of the bit array. This defines the starting bit as bit 0 in element 2, data file 3.
- **58** This is the length of the bit array, 58 bits. Note that the bits "left over" in element 5 are unusable.

You can program as many bit arrays as you like in a bit file. Be careful that they do not overlap.

Sequencer Instructions

The following figure shows a user-defined file within bit data file 3. For this particular user-defined file, you would enter the following parameters when programming the instruction:



Address of the user-defined file is #B3:4.

Length of the file is 6 elements beyond the starting address (elements labeled 0-6 in the diagram).

- **#B3:4** The address of the file. This defines the starting element as element 4, bit file 3.
- **6** The specified length of the file, 6 elements *beyond* the starting address (totals 7 elements).

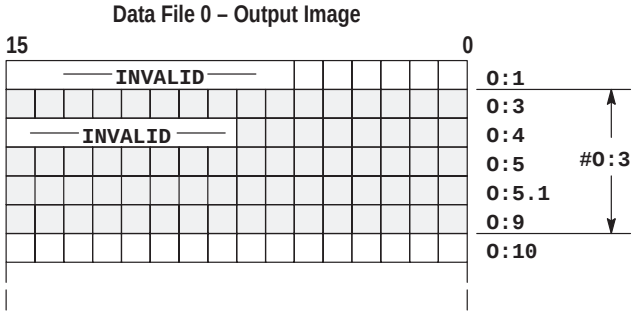
You can use user-defined integer files or bit files with sequencer instructions, depending on the application. You can program as many files as you like within another file. However, be careful that the files do not overlap.

Copy File and Fill File Instructions

These instructions manipulate user-defined files. The files are used as source or destination parameters in Copy File or Fill File instructions. Files can be Output, Input, Status, Bit, Timer, Counter, Control, or Integer files. Two examples are shown in the following figure. Note that the file length is the specified number of elements of the destination file; this differs from the file length specification for sequencer instructions.

The following figure shows a user-defined file within Data File 0- Output Image. We used this particular data file configuration in regard to I/O addressing on page F–12. Here, we are defining a file 5 elements long.

Note that for the output file (and the input file as well), an element is always one word, referenced as the slot and word taken together. For example, element **O:3.0** refers to output file, slot 3, word 0. This defaults to **O:3**, where word 0 is implied.



File #O:3 shown above is 5 elements long: Elements 3, 4, 5, 5.1, 9

Numeric Constants

You can enter numeric constants directly into many of the instructions you program. The range of values for most instructions is –32,768 through +32,767. These values can be displayed or entered in several radixes. The radixes that can be *displayed* are:

- Integer
- Binary
- ASCII
- Hexadecimal

When *entering* values into an instruction or data table element, you can specify the radix of your entry using the "&" special operator. The radixes that can be used to enter data into an instruction or data table element are:

- Integer (&N)
- Binary (&B)
- ASCII (&A)
- Hexadecimal (&H)
- BCD (&D)
- Octal (&O)

Numeric constants are used in place of data file elements. They cannot be manipulated by the user program. You must enter the offline program editor to change the value of a constant.

M0 and M1 Data Files – Specialty I/O Modules

M0 and M1 files are data files that reside in specialty I/O modules only. There is no image for these files in the processor memory. The application of these files depends on the function of the particular specialty I/O module. For some modules, the M0 file is regarded as a module output file and the M1 file is regarded as a module input file. In any case, both M0 and M1 files are considered read/write files by the SLC 5/02 and higher processors.

M0 and M1 files can be addressed in your ladder program and they can also be acted upon by the specialty I/O module-independent of the processor scan. It is important that you keep the following in mind in creating and applying your ladder logic:

Note *During the processor scan, M0 and M1 data can be changed* by the processor *according to ladder diagram instructions addressing the M0 and M1 files. During the same scan, the* specialty I/O module *can change M0 and M1 data,* ***independent*** *of the rung logic applied during the scan.*

Addressing M0–M1 Files

The addressing format for M0 and M1 files is below:

Mf:e.s/b

Where **M = module**
f = file type (0 or 1)
e = slot (1–30)
s = word (0 to max. supplied by module)
b = bit (0–15)

Restrictions on Using M0-M1 Data File Addresses

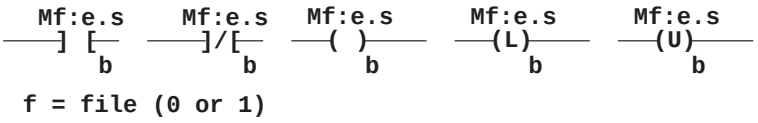
M0 and M1 data file addresses can be used in all instructions except the OSR instruction and the instruction parameters noted below:

Instruction	Parameter (uses file indicator #)
BSL, BSR	File (bit array)
SQO, SQC, SQL	File (sequencer file)
LFL, LFU	LIFO (stack)
FFL, FFU	FIFO (stack)

Monitoring Bit Addresses

SLC 5/02 and Higher Processors with M0 and M1 Monitoring Disabled

When you monitor a ladder program in the run or test mode, the following bit instructions, addressed to an M0 or M1 file, are indicated as false regardless of their actual true/false logical state.



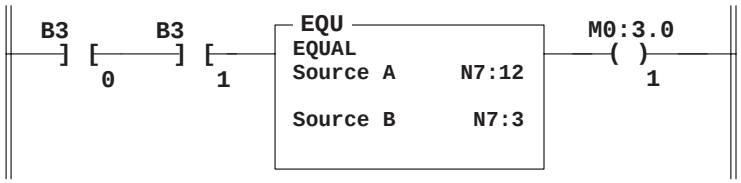
When you are monitoring the ladder program in the run or test mode, the APS or HHT display does not show these instructions as being true when the processor evaluates them as true.

SLC 5/03 and Higher Processors with M0 and M1 Monitoring Enabled

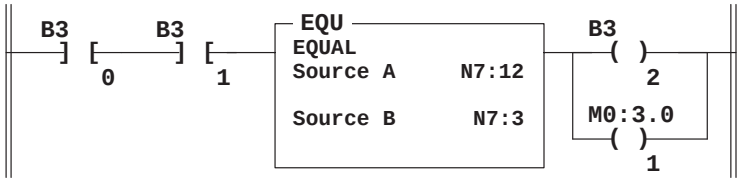
The SLC 5/03 and higher processors allow you to monitor the actual state of each addressed M0/M1 address (or data table). The highlighting appears normal when compared to the other processor data file. The SLC 5/03's performance is degraded to the degree of M0/M1 referenced screen data. For example, if your screen has only one M0/M1 element, degradation is minimal. If your screen has 69 M0/M1 elements, degradation is significant.

If you need to show the state of the M0 or M1 addressed bit, you can transfer the state to an internal processor bit. This is illustrated in the following figure, where an internal processor bit is used to indicate the true/false state of a rung.

This rung will not show its true rungstate because the EQU instruction is always shown as true and the M0 instruction is always shown as false.



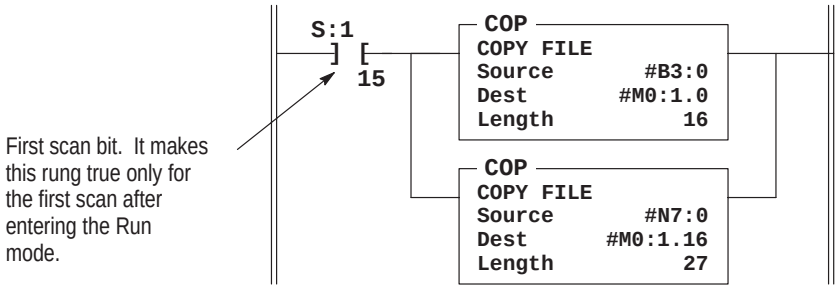
OTE instruction B3/2 has been added to the rung. This instruction shows the true or false state of the rung.



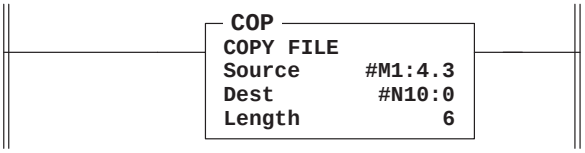
Transferring Data Between Processor Files and M0 or M1 Files

As pointed out earlier, the processor does not contain an image of the M0 or M1 file. As a result, you must edit and monitor M0 and M1 file data via instructions in your ladder program. For example, you can copy a block of data from a processor data file to an M0 or M1 data file or vice versa using the COP instruction in your ladder program.

The COP instructions below copy data from a processor bit file and integer file to an M0 file. Suppose the data is configuration information affecting the operation of the specialty I/O module.



The COP instruction that follows copies data form an M1 data file to an integer file. This technique is used to monitor the contents of an M0 or M1 data file indirectly, in a processor data file.

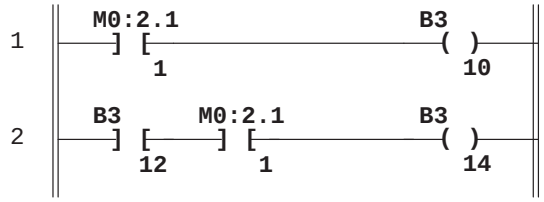


Access Time

During the program scan, the processor must access the specialty I/O card to read/write M0 or M1 data. This access time must be added to the execution time of each instruction referencing M0 or M1 data. Refer to appendix C in this manual for the access times and an example.

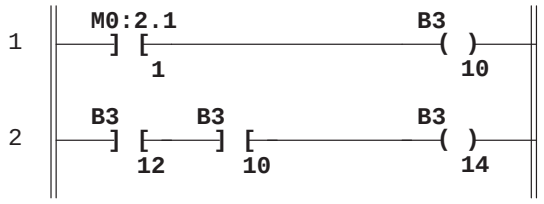
Minimizing the Scan Time

You can keep the processor scan time to a minimum by economizing on the use of instructions addressing the M0 or M1 files. For example, XIC instruction M0:2.1/1 is used in rungs 1 and 2 of the figure below, adding approximately 2 ms to the scan time if you are using a Series B processor.



XIC instructions in rungs 1 and 2 are addressed to the M0 data file. Each of these instructions adds approximately 1 ms to the scan time (Series B processor).

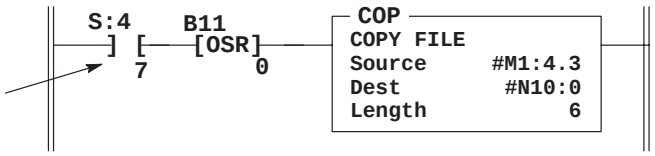
In the equivalent rungs of the following figure, XIC instruction M0:2.1/1 is used only in rung 1, reducing the scan time by approximately 1 ms.



These rungs provide equivalent operation to those of figure A by substituting XIC instruction B3/10 for XIC instruction M0:2.1/1 in rung 2. Scan time is reduced by approximately 1 ms (Series B processor).

The following figure illustrates another economizing technique. The COP instruction addresses an M1 file, adding approximately 4.29 ms to the scan time if you are using a Series B processor. Scan time economy is realized by making this rung true only periodically, as determined by clock bit S:4/8. (Clock bits are discussed in appendix B in this manual.) A rung such as this might be used when you want to monitor the contents of the M1 file, but monitoring need not be on a continuous basis.

S:4/7 causes the #M1:4.3 file to update the #N10:0 file every 2.56 seconds.



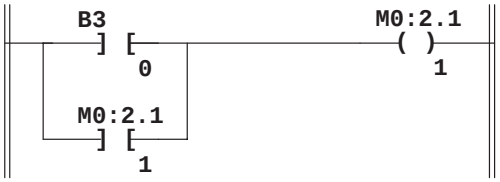
Capturing M0-M1 File Data

The first two ladder diagrams in the last section illustrate a technique allowing you to capture and use M0 or M1 data as it exists at a particular time. In the first figure, bit M0:2.1/1 could change state between rungs 1 and 2. This could interfere with the logic applied in rung 2. The second figure avoids the problem. If rung 1 is true, bit B3/10 captures this information and places it in rung 2.

In the second example of the last section, a COP instruction is used to monitor the contents of an M1 file. When the instruction goes true, the 6 words of data in file #M1:4.3 is captured as it exists at that time and placed in file #N10.0.

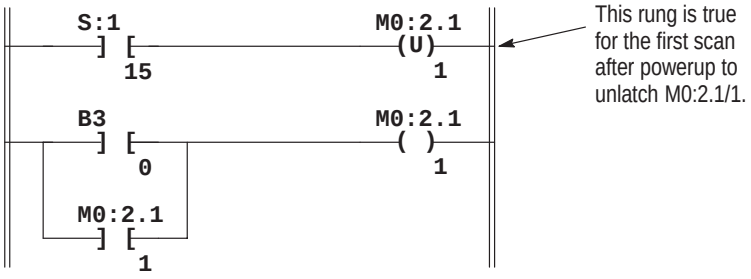
Specialty I/O Modules with Retentive Memory

Certain specialty I/O modules retain the status of M0-M1 data after power is removed. See your specialty I/O module user's manual. This means that an OTE instruction having an M0 or M1 address remains on if it is on when power is removed. A "hold-in" rung as shown below will not function as it would if the OTE instruction were non-retentive on power loss. If the rung is true at the time power is removed, the OTE instruction latches instead of dropping out; when power is again applied, the rung is evaluated as true instead of false.



When used with a speciality I/O module having retentive outputs, this rung can cause unexpected start-up on powerup.

You can achieve non-retentive operation by unlatching the retentive output with the first pass bit at powerup:



G Data Files – Specialty I/O Modules

Some specialty I/O modules use G (confiGuration) files (indicated in the specific specialty I/O module user's manual). These files can be thought of as the software equivalent of DIP switches.

The content of G files is accessed and edited offline under the I/O Configuration function. You cannot access G files under the Monitor File function. Data you enter into the G file is passed on to the specialty I/O module when you download the processor file and enter the REM Run or any one of the REM Test modes.

The following figure illustrates the three G file data formats that you can select. Word addresses begin with the file identifier G and the slot number you have assigned to the specialty I/O module. In this case, the slot number is 1. Sixteen words have been created (addresses G1:0 through G1:15).

16-word G file, I/O slot 1, decimal format

```
address    0      1      2      3      4      5      6      7      8      9
G1:0     xxxx     0      0      0      0      0      0      0      0      0
G1:10       0      0      0      0      0      0
```

16-word G file, I/O slot 1, hex/bcd format

```
address    0      1      2      3      4      5      6      7      8      9
G1:0     xxxx   0000   0000   0000   0000   0000   0000   0000   0000   0000
G1:10    0000   0000   0000   0000   0000   0000
```

16-word G file, I/O slot 1, binary format

```
address  15        data        0
G1:0      xxxx  xxxx  xxxx  xxxx
G1:1      0000  0000  0000  0000
G1:2      0000  0000  0000  0000
G1:3      0000  0000  0000  0000
G1:4      0000  0000  0000  0000
G1:5      0000  0000  0000  0000
G1:6      0000  0000  0000  0000
G1:7      0000  0000  0000  0000
G1:8      0000  0000  0000  0000
G1:9      0000  0000  0000  0000
G1:10     0000  0000  0000  0000
G1:11     0000  0000  0000  0000
G1:12     0000  0000  0000  0000
G1:13     0000  0000  0000  0000
G1:14     0000  0000  0000  0000
G1:15     0000  0000  0000  0000
```

Editing G File Data

Data in the G file must be edited according to your application and the requirements of the specialty I/O module. You edit the data offline under the I/O configuration function only. With the decimal and hex/bcd formats, you edit data at the word level:

- G1:1 = 234 (decimal format)
  G1:1 = 00EA (hex/bcd format)
- With the binary format, you edit data at the bit level:
  G1/19 = 1

Note *Word 0 of the G file is configured automatically by the processor according to the particular specialty I/O module. Word 0 cannot be edited.*

G Number Systems

This appendix:

- covers binary and hexadecimal numbers
- explains the use of a hex mask to filter data in certain programming instructions

Binary Numbers

The processor memory stores 16-bit binary numbers. As indicated in the following figure, each position in the number has a decimal value, beginning at the right with 2^0 and ending at the left with 2^{15}.

Each position can be 0 or 1 in the processor memory. A 0 indicates a value of 0; a 1 indicates the decimal value of the position. The equivalent decimal value of the binary number is the sum of the position values.

Positive Decimal Values

The far left position will always be 0 for positive values. As indicated in the figure, this limits the maximum positive decimal value to 32767. All positions are 1 except the far left position.

Other examples:

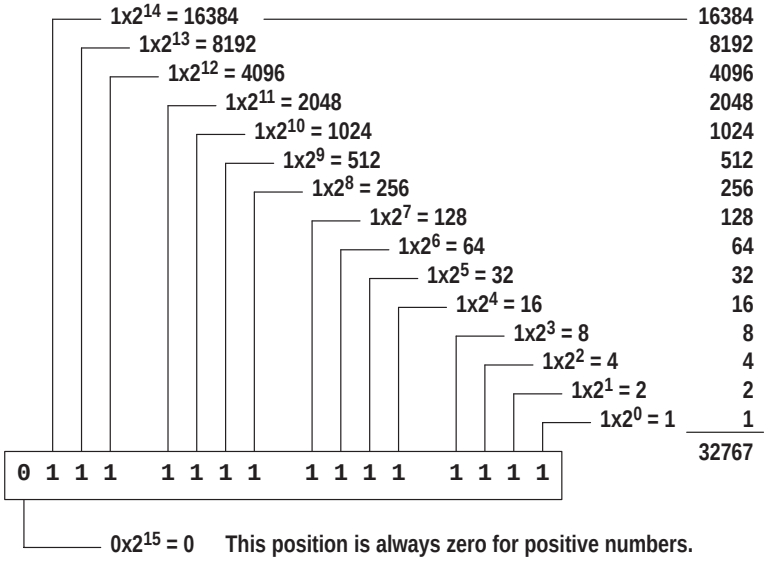
0000 1001 0000 1110 $= 2^{11}+2^8+2^3+2^2+2^1$

$= 2048+256+8+4+2 = 2318$

0010 0011 0010 1000 $= 2^{13}+2^9+2^8+2^5+2^3$

$= 8192+512+256+32+8$

$= 9000$



Negative Decimal Values

The 2s complement notation is used. The far left position is always 1 for negative values. The equivalent decimal value of the binary number is obtained by subtracting the value of the far left position, 32768, from the sum of the values of the other positions. In the following figure, the value is 32767 – 32768 = –1. All positions are 1.

Another example:

1111 1000 0010 0011 =

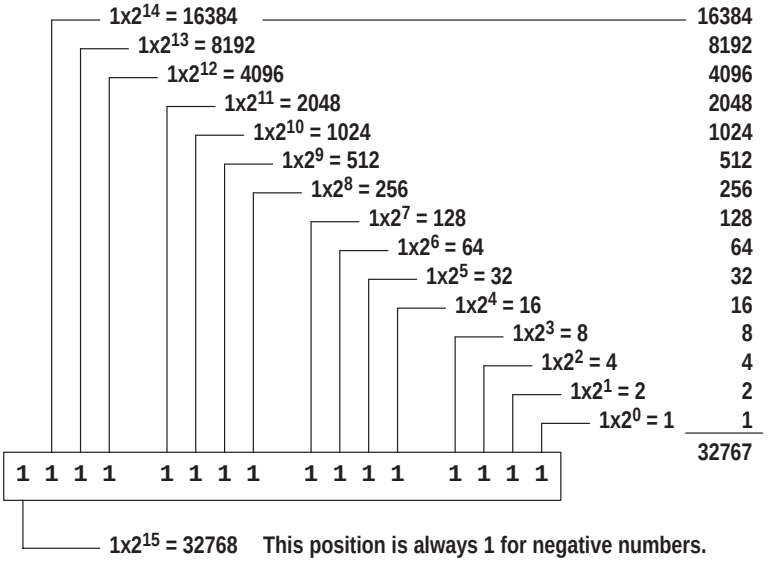
$(2^{14}+2^{13}+2^{12}+2^{11}+2^{5}+2^{1}+2^{0}) - 2^{15} =$

(16384+8192+4096+2048+32+2+1) – 32768 =

30755 – 32768 = –2013.

An often easier way to calculate a value is to locate the last 1 in the string of 1s beginning at the left, and subtract its value from the total value of positions to the right of that position. For example,

1111 1111 0001 1010 = $(2^{4}+2^{3}+2^{1}) - 2^{8}$ = (16+8+2) – 256 = –230.

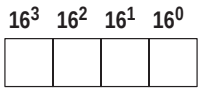


Hexadecimal Numbers

Hexadecimal numbers use single characters with equivalent decimal values ranging from 0 to 15:

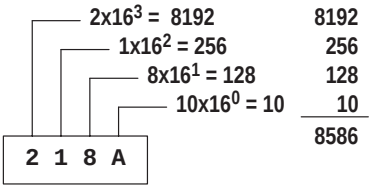
HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

The position values of hexadecimal numbers are powers of 16, beginning with 16^0 at the right:

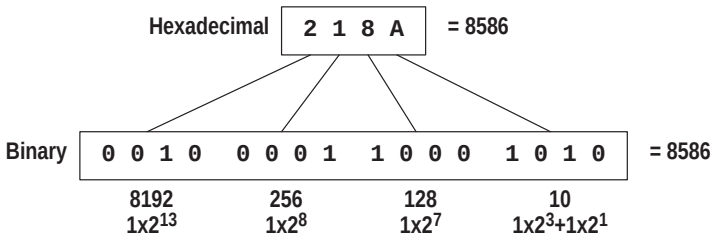


Example

Hexadecimal number 218A has a decimal equivalent value of 8586:

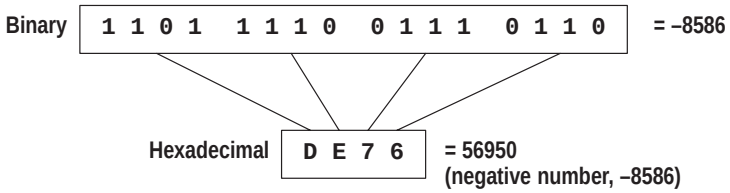


Hexadecimal and binary numbers have the following equivalence:



Example

Decimal number –8586 in equivalent binary and hexadecimal forms:

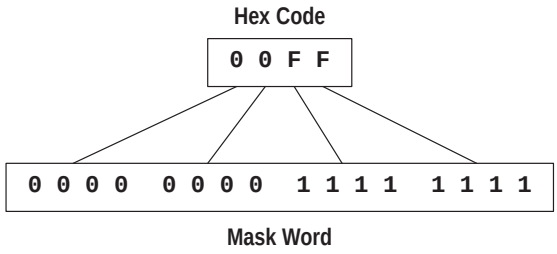


Hexadecimal number DE76 = $13x16^3+14x16^2+7x16^1+6x16^0$ = 56950. We know this is a negative number because it exceeds the maximum positive value of 32767. To calculate its value, subtract 16^4 (the next higher power of 16) from 56950: 56950 – 65536 = –8586.

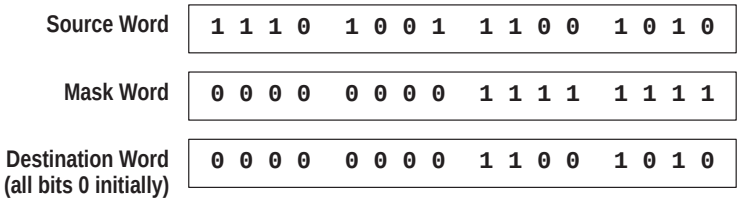
Hex Mask

This is a 4-character code, entered as a parameter in SQO, SQC, and other instructions to exclude selected bits of a word from being operated on by the instruction. The hexadecimal values are used in their binary equivalent form, as indicated in the figure below. The figure also shows an example of a hexadecimal code and the corresponding mask word.

Hex Value	Binary Value
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
A	1010
B	1011
C	1100
D	1101
E	1110
F	1111



Bits of the mask word that are set (1) will pass data from a source to a destination. Reset bits (0) will not. In the example below, data in bits 0–7 of the source word is passed to the destination word. Data in bits 8–15 of the source word is not passed to the destination word.



Binary Floating-Point Arithmetic

The SLC 5/03 Series B, SLC 5/04, and SLC 5/05 processors support the use of floating-point. Use floating-point when you want to manipulate numbers outside of the range –32768 to +32767 or for a resolution finer than one unit. For example, 2.075. Floating-point arithmetic does not support unnormalized, Not a Number (NaN), and infinity. The valid range for a floating-point number is ±3.402824 x 10^{38} to ±1.1754944 x 10^{-38}.

The following example shows the representation of a floating-point number using the IEEE 754 standard for Single precision floating-point.

The following is the spatial representation of the 32 bits in the register.

where:
- s = sign
- x = exponent
- m = mantissa

When converting to floating-point arithmetic, the following must occur:

1. The sign bit must be set. If the number is positive, then the sign bit is 0 or Off. If the number is negative, then the sign bit is 1 or On.

2. The exponent must be normalized. Do this by *always* adding +127 to the exponent.

3. The mantissa must be normalized. For example, the binary value of 1010.01 equals 1.01001

4. The fraction must be extracted from the mantissa. For example, the fractional part of 1.01001 is .01001.

The 32–bit floating-point representation of 10.25 decimal equals:

0 10000010 01001000000000000000000

H Application Example Programs

This appendix is designed to illustrate various instructions described previously in this manual. Application example programs include:

- paper drilling machine using most of the instructions
- time driven sequencer using TON and SQO instructions
- event driven sequencer using SQC and SQO instructions
- on/off circuit using basic, program flow, and application specific instructions
- interfacing with enhanced bar code decoders over DH-485

Because of the variety of uses for this information, the user of and those responsible for applying this information must satisfy themselves as to the acceptability of each application and use of the program. In no event will Allen–Bradley Company be responsible or liable for indirect or consequential damages resulting from the use of application of this information.

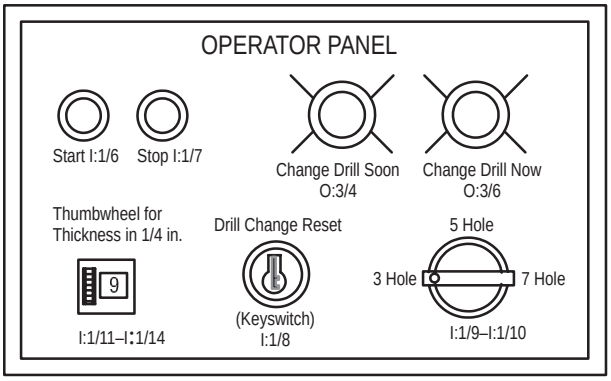
The illustrations, charts, and examples shown in this appendix are intended solely to illustrate the principles of the controller and some of the methods used to apply them. Particularly because of the many requirements associated with any particular installation, Allen–Bradley Company cannot assume responsibility or liability for actual use based upon the illustrative uses and applications.

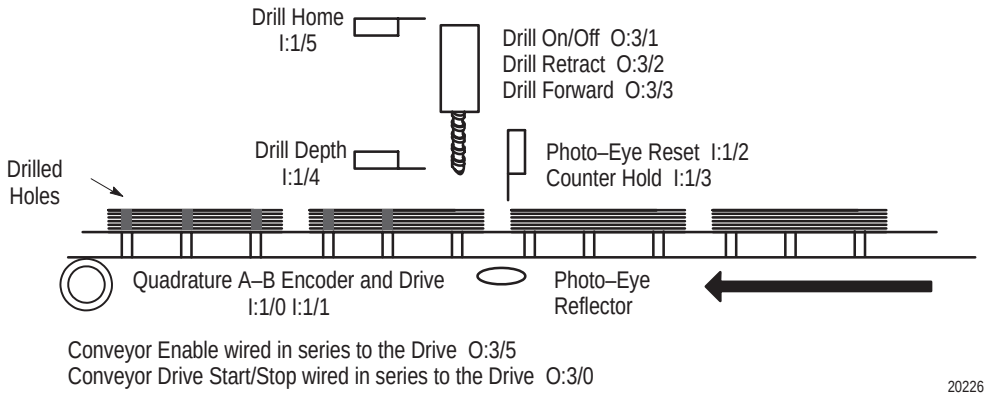
Paper Drilling Machine Application Example

For a detailed explanation of:

- XIC, XIO, OTE, RES, OTU, OTL, and OSR instructions, see chapter 1.
- EQU and GEQ instructions, see chapter 2.
- CLR, ADD, and SUB instructions, see chapter 3.
- MOV and FRD instructions, see chapter 4.
- JSR and RET instructions, see chapter 5.
- INT and SQO instructions, see chapter 11.
- HSC, HSL, and RAC instructions, see chapter 7.

This machine can drill 3 different hole patterns into bound manuals. The program tracks drill wear and signals the operator that the bit needs replacement. The machine shuts down if the signal is ignored by the operator.





Paper Drilling Machine Operation Overview

Undrilled books are placed onto a conveyor taking them to a single spindle drill. Each book moves down the conveyor until it reaches the first drilling position. The conveyor stops moving and the drill lowers and drills the first hole. The drill then retracts and the conveyor moves the same book to the second drilling position. The drilling process is repeated until there are the desired holes per book.

Drill Mechanism Operation

When the operator presses the start button, the drill motor turns on. After the book is in the first drilling position, the conveyor subroutine sets a drill sequence start bit, and the drill moves toward the book. When the drill has drilled through the book, the drill body hits a limit switch and causes the drill to retract up out of the book. When the drill body is fully retracted, the drill body hits another limit switch indicating that it is in the home position. Hitting the second limit switch unlatches the drill sequence start bit and causes the conveyor to move the book to the next drilling position.

Conveyor Operation

When the start button is pressed, the conveyor moves the books forward. As the first book moves close to the drill, the book trips a photo-eye sensor. This tells the machine where the leading edge of the book is. Based on the position of the selector switch, the conveyor moves the book until it reaches the first drilling position. The drill sequence start bit is set and the first hole is drilled. The drill sequence start bit is now unlatched and the conveyor moves the same book to the second drilling position. The drilling process is repeated until there are the desired holes per book. The machine then looks for another book to break the photo-eye beam and the process is repeated. The operator can change the number of drilled holes by changing the selector switch.

Drill Calculation and Warning

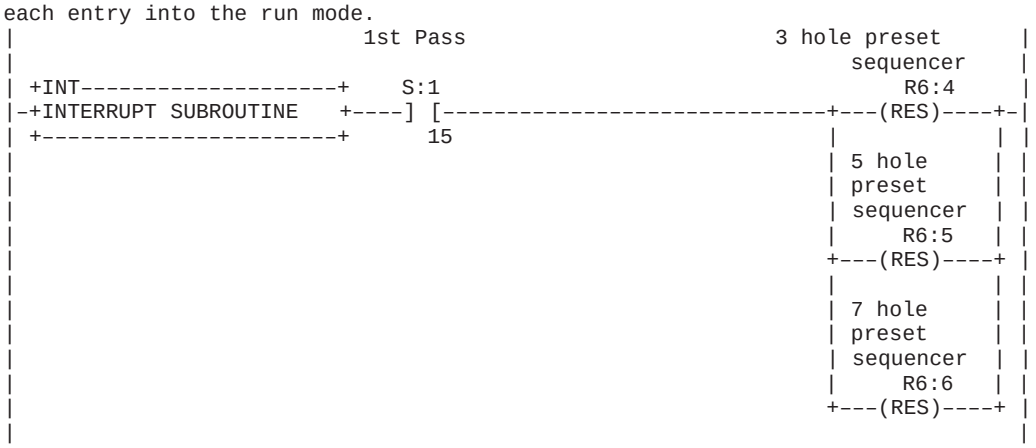
The program tracks the number of holes drilled and the number of inches of material that have been drilled through using a thumbwheel. The thumbwheel is set to the thickness of the book per 1/4 inch. (If the book is 1 1/2 inches thick, the operator would set the thumbwheel to 6.) When 25,000 inches have been drilled, the Change Drill Soon pilot light turns on. When 25,500 inches have been drilled, the Change Drill Soon pilot light flashes. When 26,000 inches have been drilled, the Change Drill Now pilot light turns on and the machine turns off. The operator changes drill bits and then resets the internal drill wear counter by turning the Drill Change Reset keyswitch.

Paper Drilling Machine Ladder Program

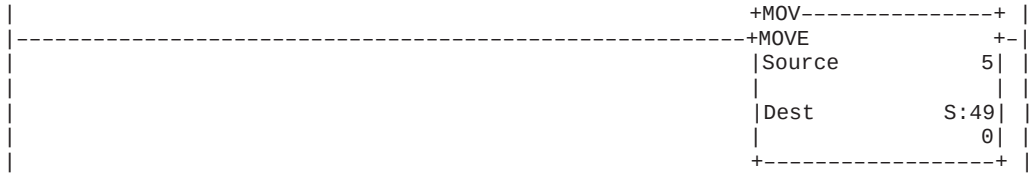
```
Rung 2:0
These rungs start the conveyer in motion when the start button is pressed.  However,
there are other conditions that must also be met before we start the conveyor.  They
are:  the drill must be in its fully retracted position (home) and the drill bit
must not be past its maximum useful life.  These rungs also stop the conveyor when
the stop button is pressed or when the drill life is exceeded.
|   START     |Drill        STOP      |change    |                  Machine    |
|   Button    |Home LS      Button    |drill bit |                    RUN      |
|                                     |NOW       |                  Latch      |
|     I:1.0      I:1.0        I:1.0      O:3.0                          B3:0   |
|-+----] [--------] [-----+----]/[--------]/[-------------------------( )-----|
| |       6          5    |       7          6                           0    |
| | Machine               |                                                     |
| |   RUN                 |                                                     |
| | Latch                 |                                                     |
| |     B3:0              |                                                     |
| +----] [----------------+                                                     |
|         0                                                                     |
Rung 2:1
| Machine                                                          Drill        |
|   RUN                                                            Motor ON     |
| Latch                                                                         |
|     B3:0                                                           O:3.0      |
|----] [-------------------------------------------------+------------( )-----+-|
|       0                                                |              1    | |
|                                                        |         Conveyor  | |
|                                                        |         Start/Stop| |
|                                                        |                   | |
|                                                        |  B3:0     O:3.0   | |
|                                                        +--[OSR]-----(L)----+ |
|                                                               1        0     |
```

```
Rung 2:2
Stop the conveyor if any conditions exist that unlatch the ”Machine RUN Latch” bit.
| Machine   |                                                          Conveyor   |
|   RUN     |                                                          Start/Stop |
| Latch     |                                                                     |
|     B3:0                                                             O:3.0    |
|----]/[-------------------------------------------------------------(U)-----|
|       0                                                                0    |
Rung 2:3
This rung calls the drill sequence subroutine. This subroutine manages the operation
of a drilling sequence and restarts the conveyer upon completion of the drilling
sequence
|                                                          +JSR---------------+ |
|----------------------------------------------------------+JUMP TO SUBROUTINE+-|
|                                                          |SBR file number  6| |
|                                                          +------------------+ |
Rung 2:4
This rung calls the subroutine that tracks the amount of wear on the current drill
bit.
|                                                          +JSR---------------+ |
|----------------------------------------------------------+JUMP TO SUBROUTINE+-|
|                                                          |SBR file number  7| |
|                                                          +------------------+ |
Rung 2:5
There is some initialization logic in the DII subroutine (file 4) that must be
executed prior to the first DII interrupt.  This rung allows the DII to be
initialized by jumping to the DII subroutine when the processor enters the RUN mode.
| 1st                                                                          |
| Pass                                                                          |
|    S:1                                                   +JSR---------------+ |
|----] [---------------------------------------------------+JUMP TO SUBROUTINE+-|
|      15                                                  |SBR file number  4| |
|                                                          +------------------+ |
Rung 2:6
|                                                                               |
|-------------------------------------+END+-------------------------------------|
|                                                                               |


FILE 3 HAS NO RUNGS
```

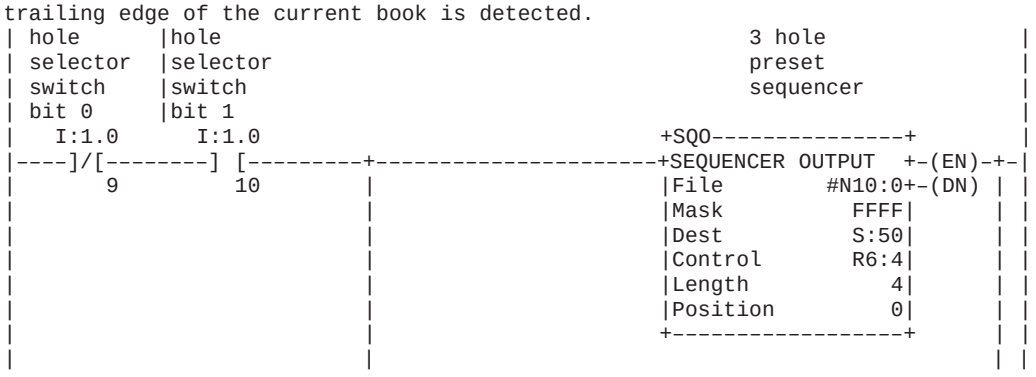
Rung 4:0
This rung resets the hole count sequencers each time the processor enters the RUN mode. This ensures that the first preset value is loaded into the DII preset at each entry into the run mode.

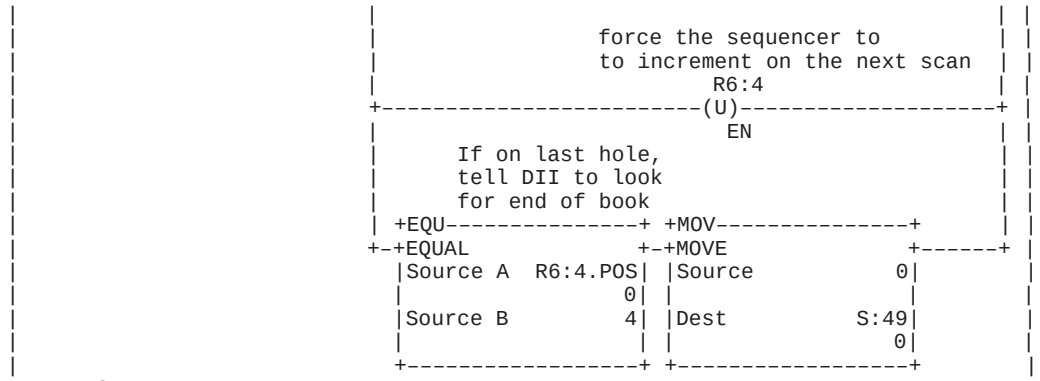


Rung 4:1
This rung tells the DII to look for the presence of a book at the photo-eye and an encoder pulse. By triggering on these two conditions, the DII only counts pulses when there is a book present, otherwise it holds at it's existing number of counts. S:49 is the DII PRESET value.

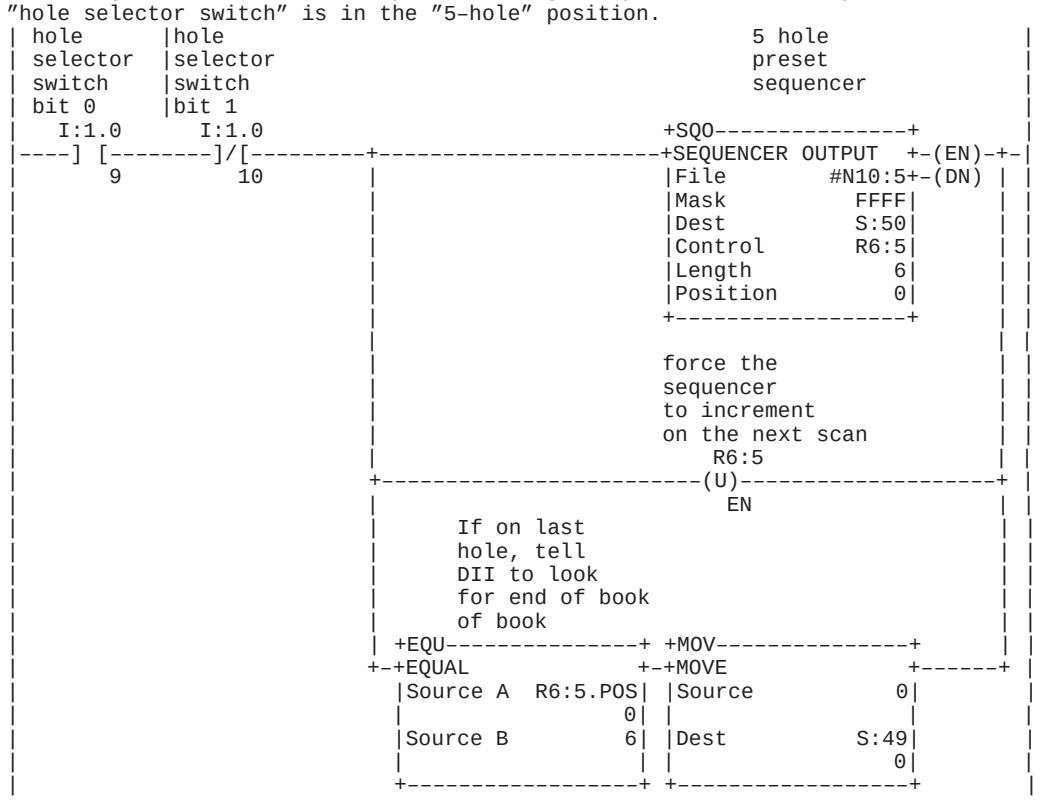


Rung 4:2
This rung keeps track of the hole number that is being drilled and loads the next correct DII preset based on the hole count. This rung is only active when the "hole selector switch" is in the "3-hole" position. The sequencer uses step 0 as a null step upon reset. It uses the last step as a "go forever" in anticipation of the "end of manual". Moving a 0 into S:49 tells the DII to trigger an interrupt when the trailing edge of the current book is detected.

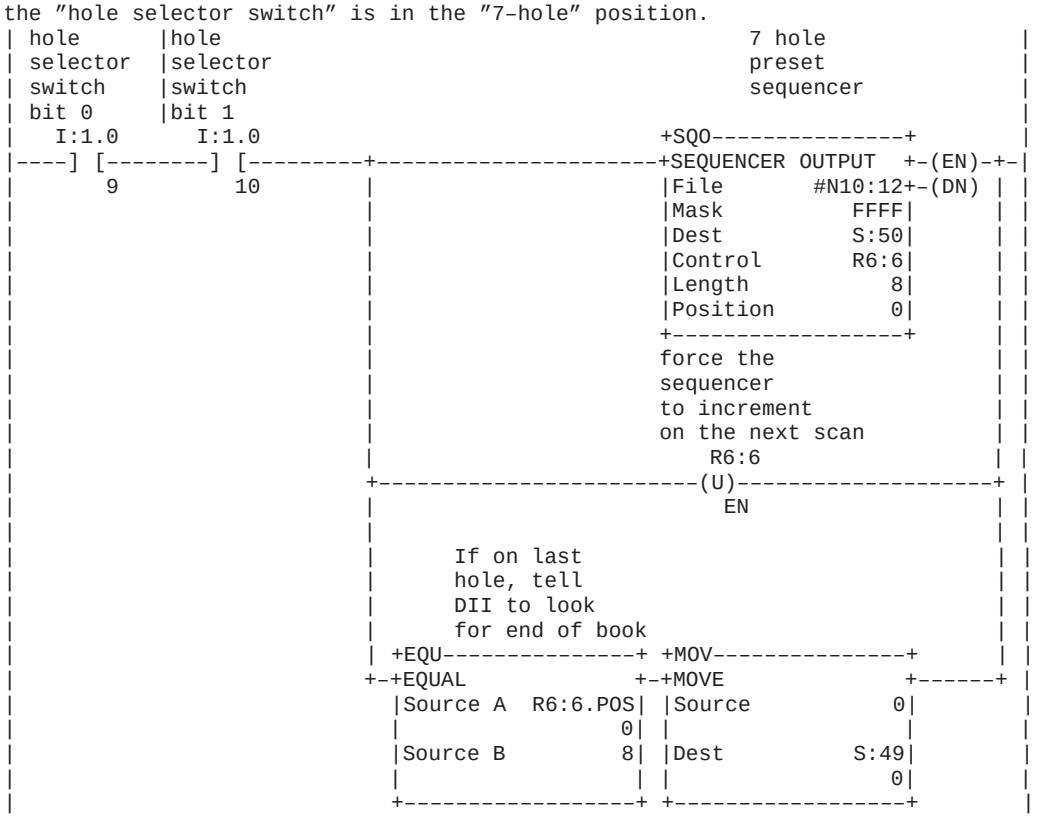




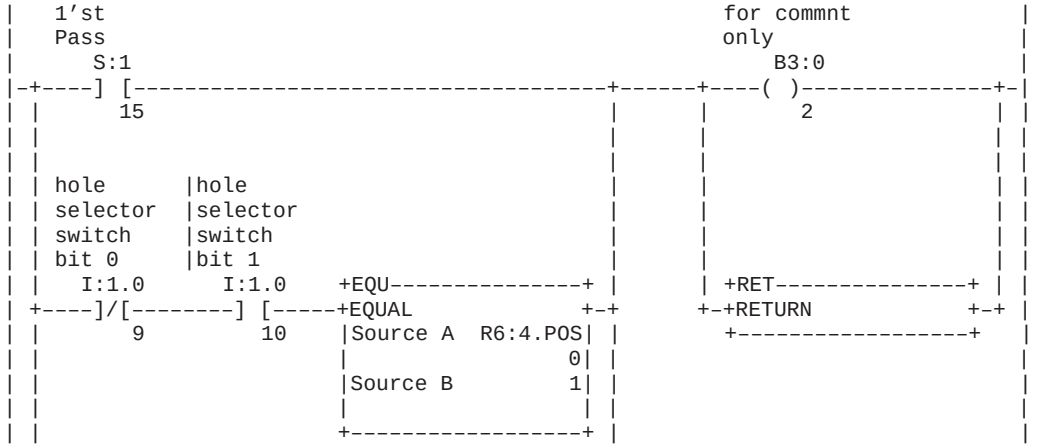
Rung 4:3
This rung is identical to the previous rung except that it is only active when the "hole selector switch" is in the "5–hole" position.

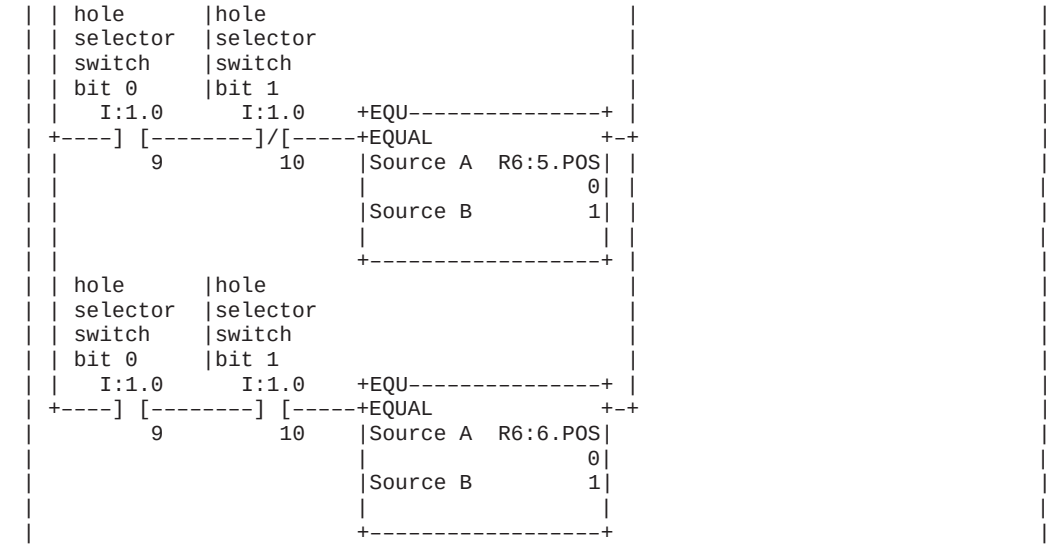


Rung 4:4
This rung is identical to the 2 previous rungs except that it is only active when the ”hole selector switch” is in the ”7–hole” position.

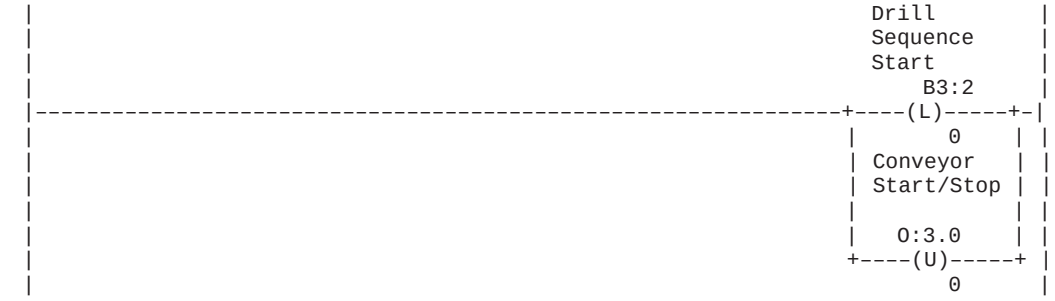


Rung 4:5
If the processor is in this subroutine either for initialization or due to sensing the trailing edge of a manual, just return and skip the logic that stops the conveyor and starts the drill sequence.

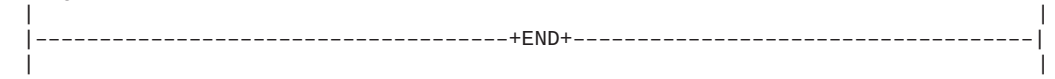




Rung 4:6
This rung stops the convoyor and signals the main program (file 2) to initiate a drilling sequence. The DRILL SEQUENCE subroutine (program file 6) resets the drill sequence start bit and sets the conveyer drive bit (O:3/0) upon completion of the drilling sequence.



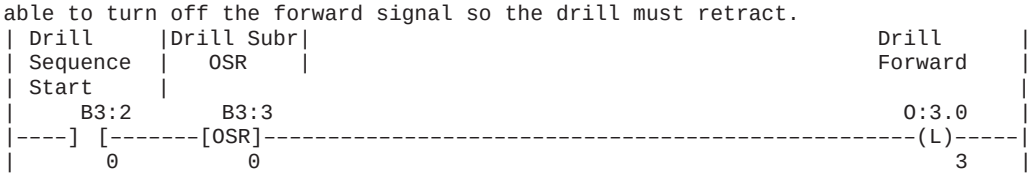
Rung 4:7



FILE 5 HAS NO RUNGS

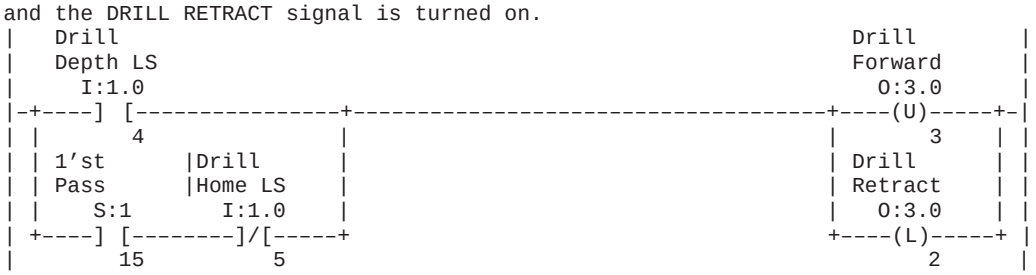
Rung 6:0

This section of ladder logic controls the up/down motion of the drill for the book drilling machine. When the conveyor positions the book under the drill, the DRILL SEQUENCE START bit is set. This rung uses that bit to begin the drilling operation. Because the bit is set for the entire drilling operation, the OSR is required to be able to turn off the forward signal so the drill must retract.



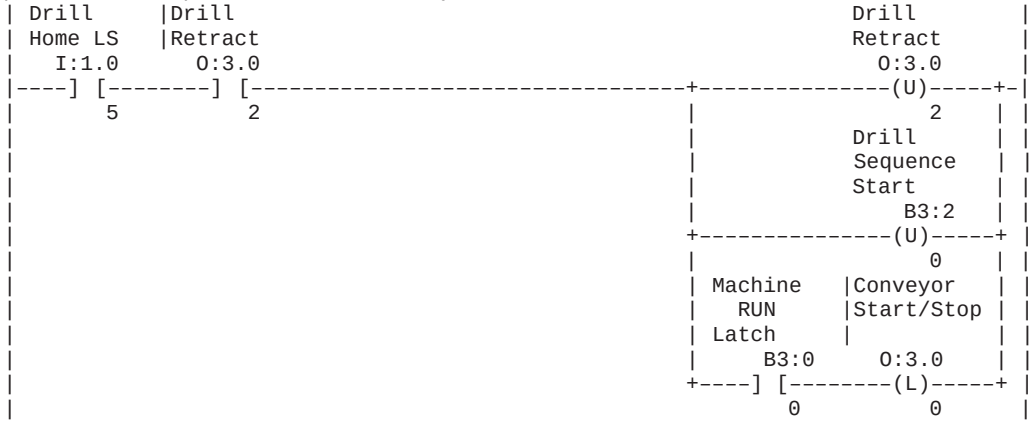
Rung 6:1

When the drill has drilled through the book, the body of the drill actuates the DRILL DEPTH limit switch. When this happens, the DRILL FORWARD signal is turned off and the DRILL RETRACT signal is turned on.

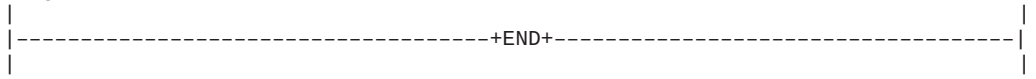


Rung 6:2

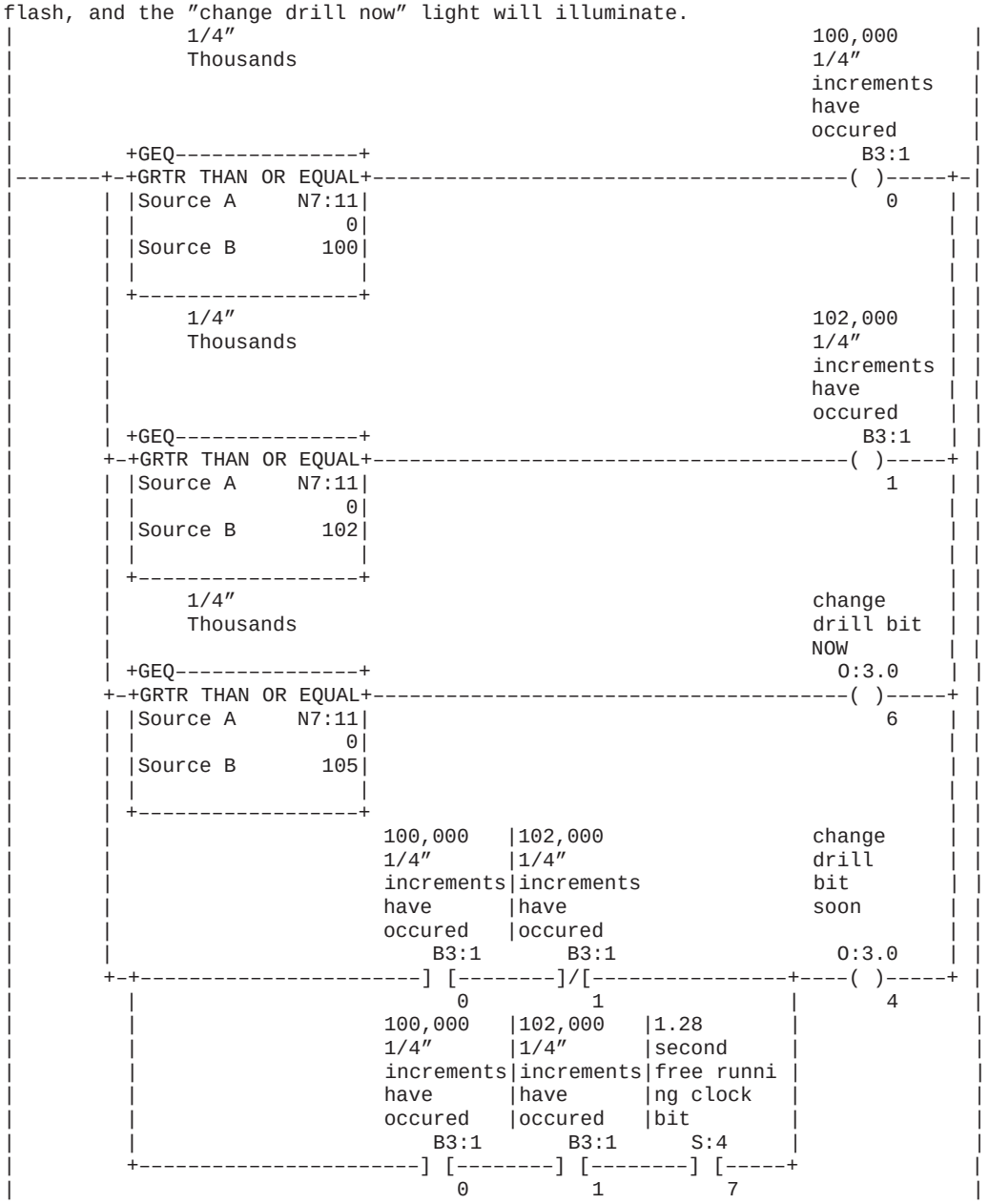
When the drill is retracting (after drilling a hole), the body of the drill will actuate the DRILL HOME limit switch. When this happens the DRILL RETRACT signal is turned off, the DRILL SEQUENCE START bit is turned off to indicate the drilling process is complete, and the conveyor is restarted.



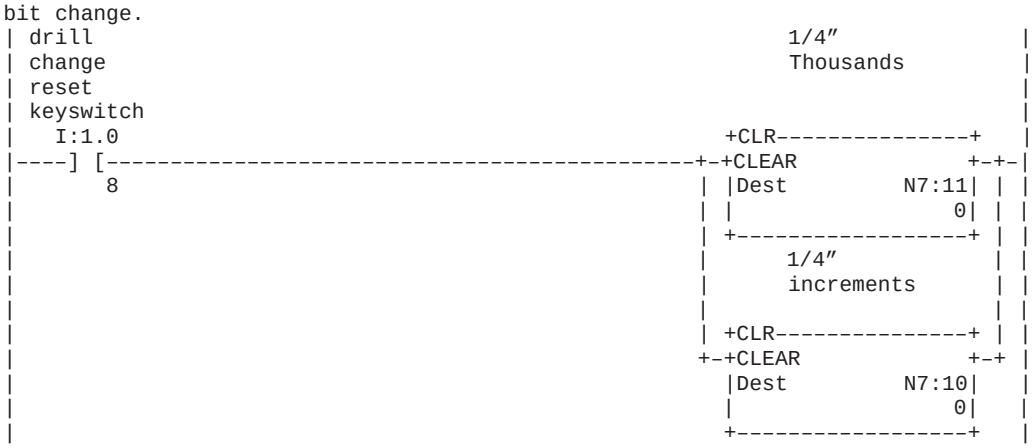
Rung 6:3



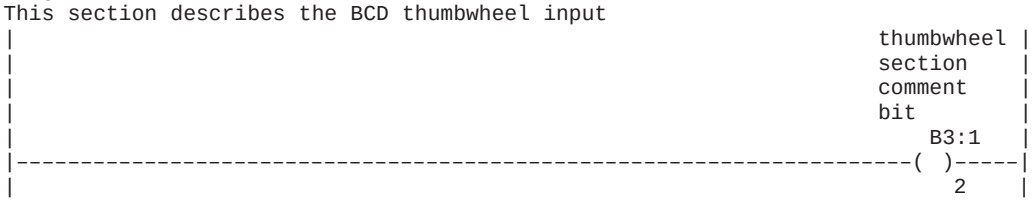
Rung 7:0
This rung examines the number of 1/4″ thousands that have accumulated over the life of the current drill bit. If the bit has drilled between 100,000–101,999 1/4″ increments of paper, then the ″change drill″ light will illuminate steady. When the value is between 102,000–103,999, then the ″change drill″ light will flash at a 1.28 second rate. When the value reaches 105,000, then the ″change drill″ light will flash, and the ″change drill now″ light will illuminate.



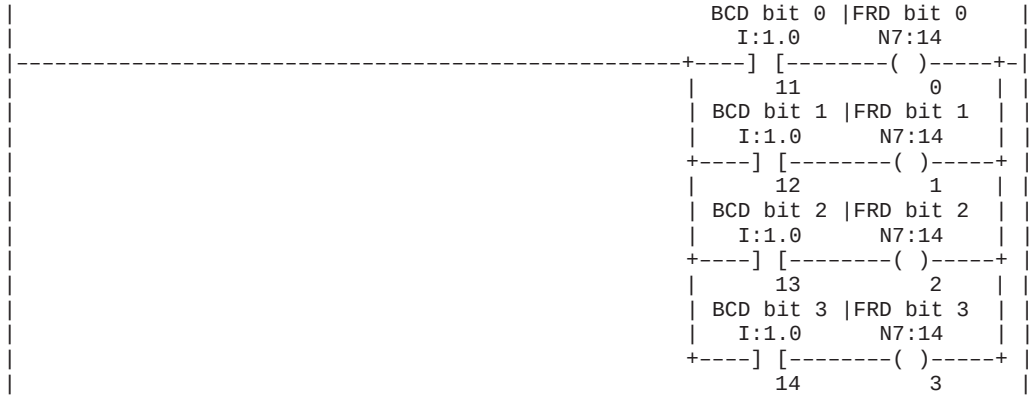
Rung 7:1
This rung resets the number of 1/4″ increments and the 1/4″ thousands when the ″drill change reset″ keyswitch is energized. This should occur following each drill bit change.



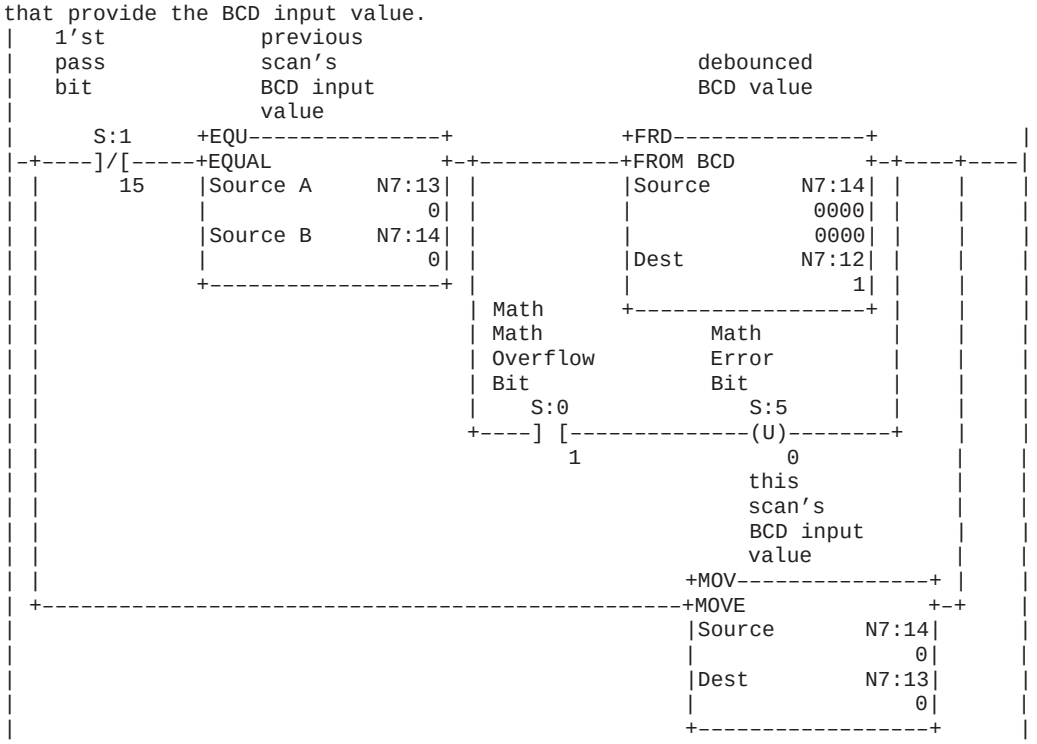
Rung 7:2
This section describes the BCD thumbwheel input



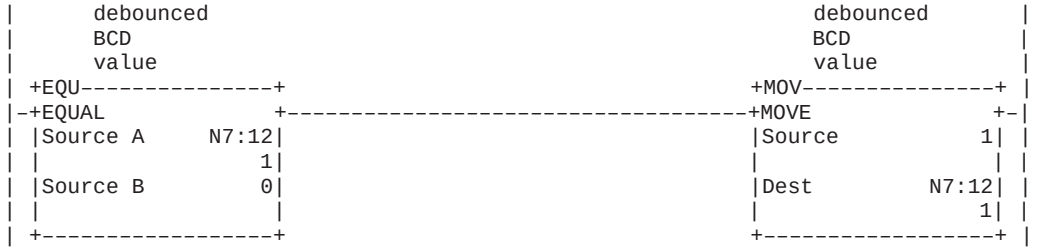
Rung 7:3
This rung moves the single digit BCD thumbwheel value into an internal Integer register. This is done to properly align the four BCD input signals prior to executing the BCD to Integer instruction (FRD). The thumbwheel is used to allow the operator to enter the thickness of the paper that is to be drilled. The thickness is entered in 1/4″ increments. This provides a range of 1/4″ to 2.25″



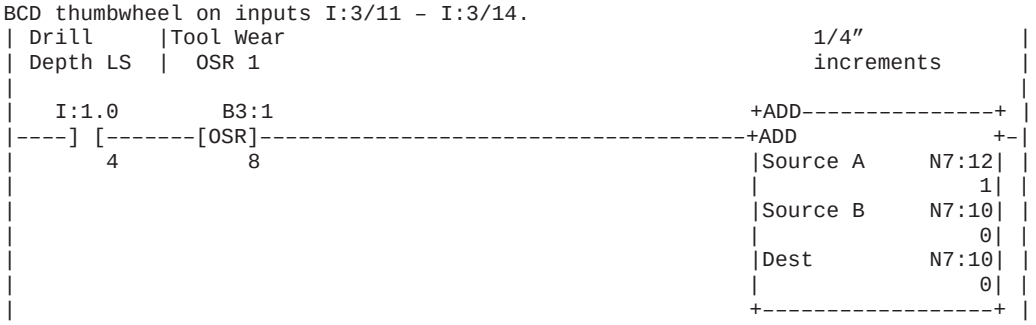
Rung 7:4
This rung converts the BCD thumbwheel value from BCD to Integer. This is done because the processor operates upon Integer values. This rung also ”debounces” the thumbwheel to ensure that the conversion only occurs on valid BCD values. Note that invalid BCD values can occur while the operator is changing the BCD thumbwheel. This is due to input filter propagation delay differences between the 4 input circuits that provide the BCD input value.



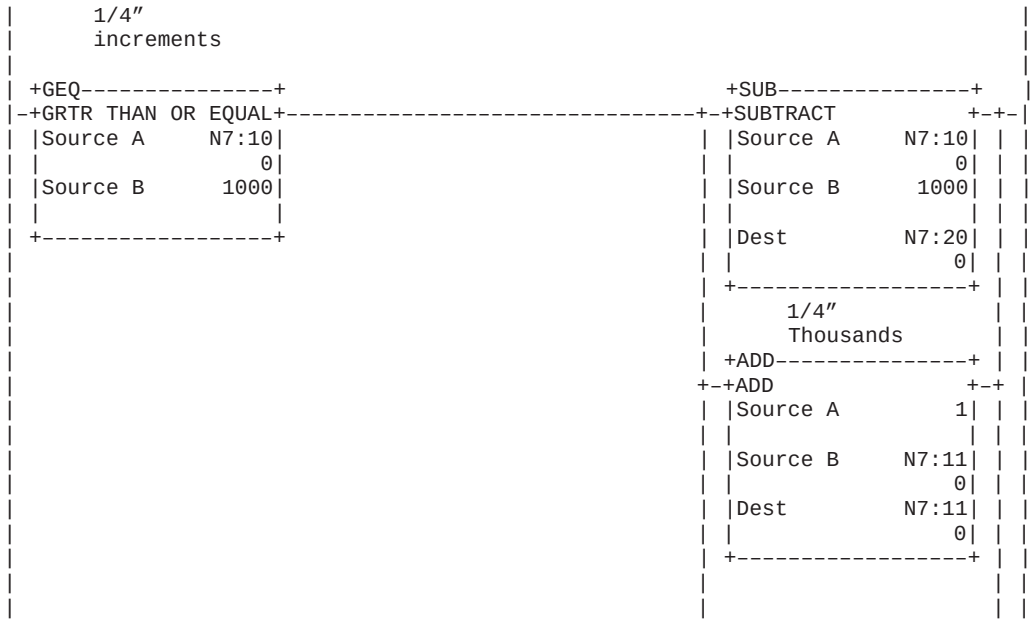
Rung 7:5
This rung ensures that the operator cannot select a paper thickness of 0. If this were allowed, the drill bit life calculation could be defeated resulting in poor quality holes due to a dull drill bit. Therefore the minimum paper thickness that will be used to calculate drill bit wear is 1/4”.



Rung 7:6
Keep a running total of how many inches of paper have been drilled with the current drill bit. Every time a hole is dril led, add the thickness (in 1/4″s) to the running total (kept in 1/4″s). The OSR is necessary because the ADD executes every time the rung is true, and the drill body would actuate the DRILL DEPTH limit switch for more than 1 program scan. Integer N7:12 is the integer–converted value of the BCD thumbwheel on inputs I:3/11 – I:3/14.



Rung 7:7
When the number of 1/4″ increments surpasses 1000, find out now many increments we are past 1000 and store in N7:20, add 1 to the total of ’1000 1/4″’ increments, and re-initialize the 1/4″ increments accumulator to how many increments were beyond 1000.



```
Rung 7:8
|                                                    |                     | |
|                                                    |        1/4”         | |
|                                                    |       increments    | |
|                                                    | +MOV--------------- | |
|                                                    +-+MOVE             +-+ |
|                                                      |Source       N7:20|   |
|                                                      |                 0|   |
|                                                      |Dest         N7:10|   |
|                                                      |                 0|   |
|                                                      +------------------+
|-------------------------------------+END+-------------------------------------|
```

```
REPORT OPTIONS SUMMARY
        Insure Valid X–Ref Info:            YES
        Graphics Mode:                      NO
        Page Width:                         80
        Page Length:                        66
        Starting File:                      2
        Ending File:                        7
        Power Rail:                         YES
        Address Comments:                   YES
        Address Display:                    YES
        Rung Comments:                      YES
        Ladder Cross Reference:             NONE
```

Time Driven Sequencer Application Example

The following application example illustrates the use of the TON and SQO instructions in a traffic signal at an intersection. The timing requirements are:

- Red light – 30 seconds
- Yellow light – 15 seconds
- Green light – 60 seconds

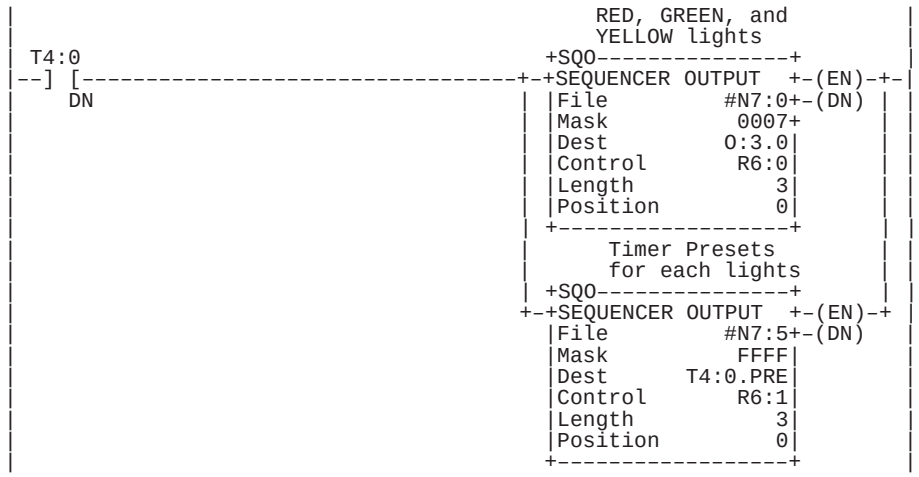
The timer, when it reaches its preset, steps the sequencer that in turn controls which traffic signal is illuminated. For a detailed explanation of:

- XIC, XIO, and TON instructions, see chapter 1.
- SQO and SQC instructions, see chapter 6.

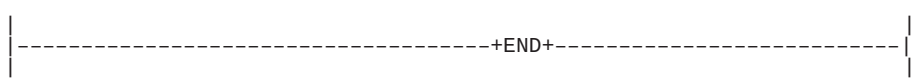
Time Driven Sequencer Ladder Program

```
Rung 2:0
The function of this rung is called a regenerative timer.  Every time the timer reaches
its preset, the DONE bit is set for one scan--this causes this rung to become FALSE for
one scan and resets the timer.  On the following scan, when this rung becomes TRUE again,
the timer begins timing.

|  Timer                                        Timer                 |
|  Enable                                                             |
|  T4:0                                    +TON---------------+       |
|---]/[-----------------------------------+TIMER ON DELAY    +-(EN)-|
|      DN                                  |Timer         T4:0+-(DN) |
|                                          |Time Base     0.01|      |
|                                          |Preset           1|      |
|                                          |Accum            0|      |
|                                          +------------------+      |
```

Rung 2:1
Controls the RED, GREEN, and YELLOW lights wired to outputs O:3/0 – O:3/2, and controls how long the regenerative timer times between each step. When this rung goes from false–to–true (by the timer reaching its preset), the first sequencer changes which traffic light is illuminated, and the second sequencer changes the preset of the timer to determine how long this next light is illuminated.



Rung 2.2



Data Files

Address	15	Data		0
N7:0	0000	0000	0000	0000
N7:1	0000	0000	0000	0100
N7:2	0000	0000	0000	0010
N7:3	0000	0000	0000	0001

Data Table

Address	Data	(Radix=Decimal)						
N7:0	0	4	2	1	0	0	6000	1500	3000

Event Driven Sequencer Application Example

The following application example illustrates how the FD (found) bit on an SQC instruction can be used to advance an SQO to the next step (position). This application program is used when a specific order of events is required to occur repeatedly. By using this combination, you can eliminate using the XIO, XIC, and other instructions. For a detailed explanation of:

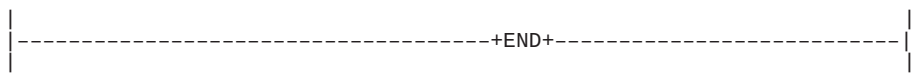
- XIC, XIO, and RES instructions, see chapter 1.
- SQO and SQC instructions, see chapter 6.

Event Driven Sequencer Ladder Program

```
Rung 2:0
Ensures that the SQO always resets to step (position) 1 each REM Run mode entry.  (This
rung actually resets the control register's position and EN enable bit to 0. Due to this
the following rung sees a false-to-true transition and asserts step (position) 1 on the
first scan.)

Eliminate this rung for retentive operation.
|  S:1                                                    R6:0     |
|--] [-------------------------------------------------------(RES)----|
|    15                                                            |
|                                                                  |
```

```
Rung 2:1
The SQC instruction and SQO instruction share the same Control Register. This is
acceptable due to the careful planning of the rungstate condition. You could cascade
(branch) many more SQO instructions below the SQO if you desired, all using the same
Control Register (R6:0 in this case).  Notice that we are only comparing Inputs 0–3 and
are only asserting Outputs 0–3 (per our Mask value).

| R6:0                                  +SQC---------------+        |
|--]/[----------------------------+-------+SEQUENCER COMPARE +-(EN)-+-|
|    FD                           |       |File         #N7:0+-(DN) | |
|                                 |       |Mask          000F+-(FD) | |
|                                 |       |Source       I:1.0|      | |
|                                 |       |Control       R6:0|      | |
|                                 |       |Length           9|      | |
|                                 |       |Position         2|      | |
|                                 |       +------------------+      | |
|                                 | R6:0  +SQO---------------+      | |
|                                 +--]/[--+SEQUENCER OUTPUT  +-(EN)-+ |
|                                      FD |File        #N7:10+-(DN)   |
|                                         |Mask           000F|      |
|                                         |Dest          O:3.0|      |
|                                         |Control       R6:0|      |
|                                         |Length           9|      |
|                                         |Position         2|      |
|                                         +------------------+      |
```

Rung 2.2

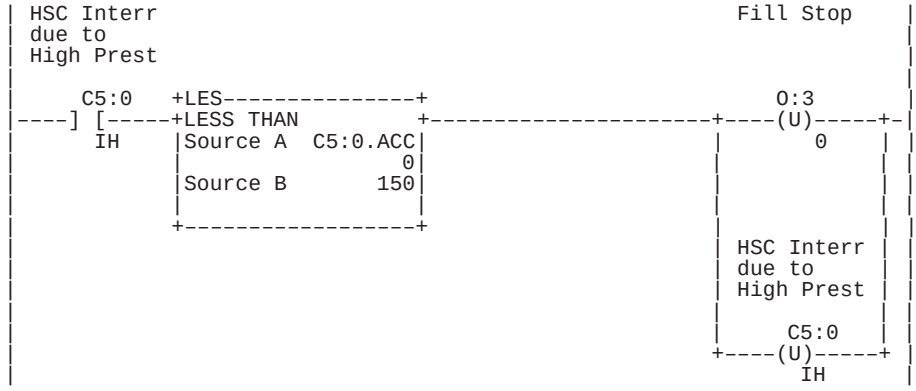


The following displays the FILE DATA for both sequencers. The SQC compare data starts at N7:0 and ends at N7:9. While the SQO output data starts at N7:10 and ends at N7:19. Please note that step 0 of the SQO is never active. The reset rung combined with the rung logic of sequencers guarantees that the sequencers always start at step 1. Both sequencers also ”roll over” to step 1. ”Roll Over” to step 1 is integral to all sequencer instructions.

SQC Compare Data

Addresses	Data	(Radix=Decimal)						
N7:0	0	1	2	3	4	5	6	7	8	9
N7:10	0	0	1	2	3	4	5	6	7	8

Rung 2:6
If the high–speed counter reached its high preset of 350 (indicates that the holding area reached maximum capacity), it would energize O:0/0, shutting down the filling operation. Before re–starting the filler, allow the packer to empty the holding area until it is about 1/3 full.



On/Off Circuit Application Example

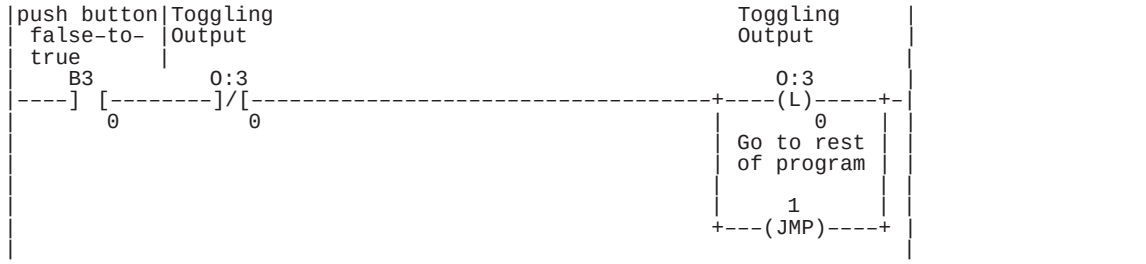
The following application example illustrates how to use an input to toggle an output either on or off. For a detailed explanation of:

- XIC, XIO, OTE, OTU, OTL, and OSR instructions, see chapter 1.
- JMP and LBL instructions, see chapter 5.

On/Off Circuit Ladder Program

Rung 2:0
Does a one–shot from the input push button to an internal bit – the internal bit is true for only one scan. This prevents toggling of the physical output in case the push button is held ”ON” for more than one scan (always the case).

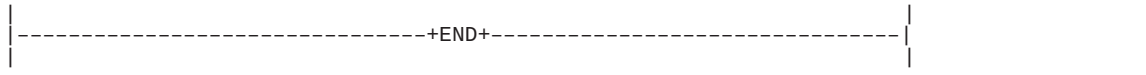
Rung 2:1
If the push button input has gone from false–to–true and the output is presently OFF, turn the output ON and jump over the following rung. If the JMP instruction was missing, the following rung would be true and would turn the output back OFF.



Rung 2:2
If the push button input has gone from false–to–true and the output is presently ON, turns the output OFF.

Rung 2:3
Contains the label corresponding to the jump instruction in rung 1. The remainder of your actual program would be placed below this rung.

Rung 2:4



Interfacing with Enhanced Bar Code Decoders Over DH-485 Network Using the MSG Instruction

The purpose of this section is to illustrate how to interface Allen-Bradley Enhanced Bar Code Decoders to SLC 5/03 and higher processors via the DH-485 network. Enhanced Bar Code Decoders act only as slave devices on this network. This means that these decoders cannot initiate the transfer of data to a host device, such as the SLC 5/03 (or higher) processor on DH-485. The SLC processor must initiate commands to a decoder and "poll" that decoder for the reply to those commands.

Processor and Decoder Operation

The Enhanced Bar Code Decoder (catalog number 2755-DS/DD, Series B), when used as a node on a DH-485 network can act as a slave only. This means that the decoder may not initiate communications to any other node on the network. Therefore, in order for a device to get bar code data from an Enhanced Bar Code Decoder on a DH-485 network, that device must send a "read" command and then "poll" the decoder for the reply with data.

The only devices capable of polling a slave device on DH-485 are the SLC 5/03 and higher processors. For the SLC 5/03 processors (1747-OS302, FRN10 or later), polling can be done via channels 0 and 1. For the SLC 5/04 processors (1747-OS401, FRN7 or later), channel 0 supports this capability.

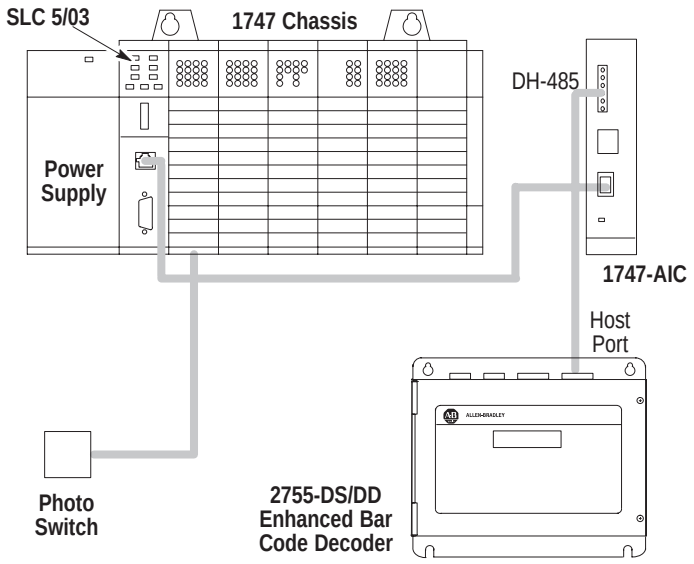
There are many ways to "trigger" bar code decoders to read a bar code label when a label is present.

- a package detect switch wired to both an SLC input module and the bar code decoder
- a package detect switch wired only to an SLC input and an SLC output then used to "trigger" the decoder
- via a software "trigger" command from the SLC processor

For this example, the software "trigger" is used. However, the basic principal is the same for all "trigger" modes.

System Set Up

In this example, a photo switch is located such that when it detects a product is in position for the bar code scanner to read a bar code label on the product, a discrete input to the SLC 5/03 processor is energized.



The 5/03 ladder program then initiates a "MSG Write" to the decoder to "trigger" the decoder to start scanning for a valid bar code label. When the decoder is scanning for a valid bar code label, it operates as shown below:

Result of Scan	Bar Code Decoder Response	Processor Response
Good Read	turns on its "Good Read" onboard output wired to the SLC processor	When one of these two inputs to the SLC are turned on, the SLC will initiate a "MSG Read" to the decoder to get the label data or no-read message data.
No-Read	turns on its "No-Read" onboard output wired to the SLC processor	

In this case, the good read output is turned on as soon as a valid read occurs, and the no-read output is turned on after the decoder has attempted to read a label for a specified amount of time and could not.

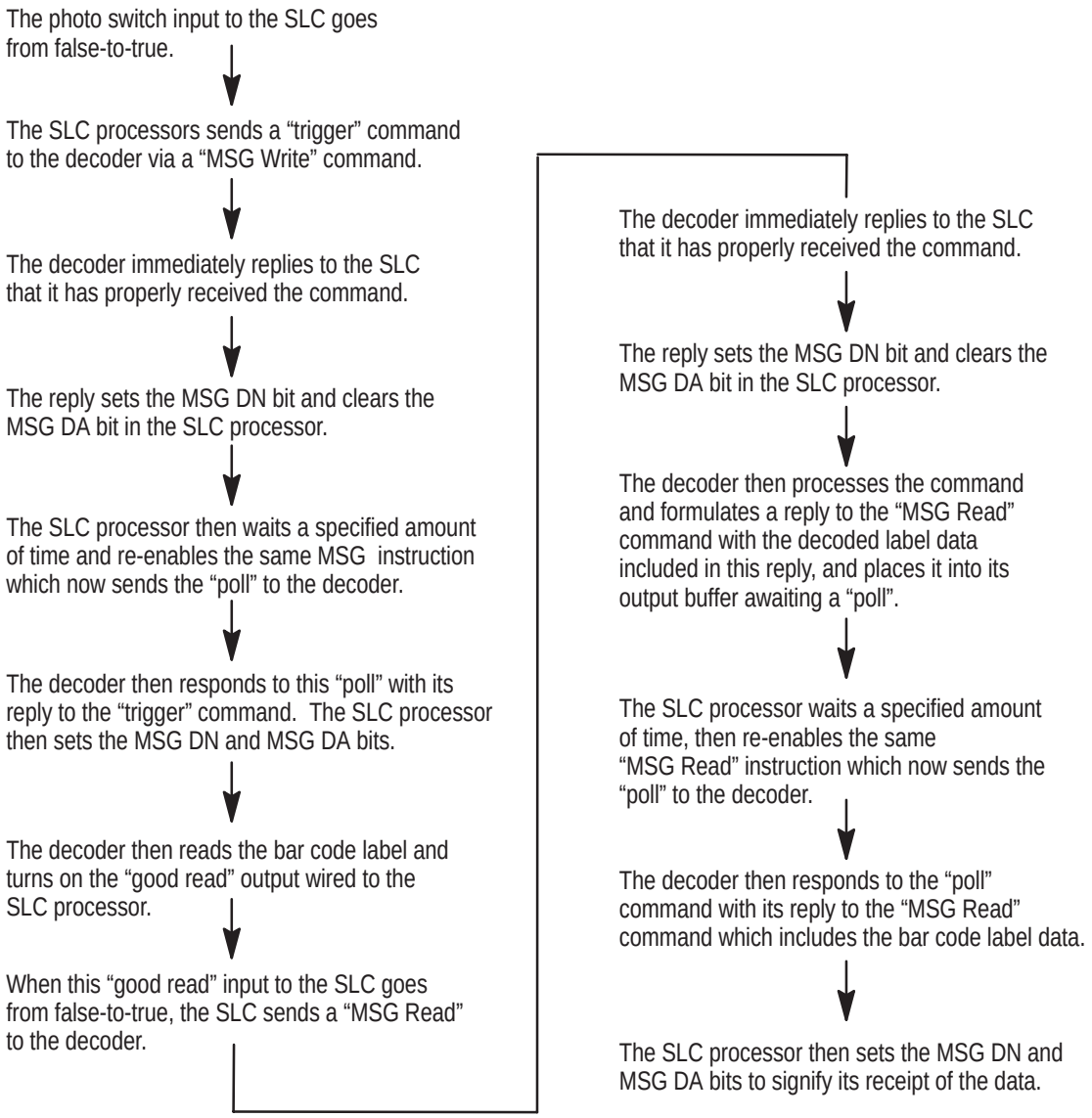
The amount of time the decoder attempts to read a label is variable and is called the "No-Read Timer". For this example it is assumed that the product is moving by the scanner and if the label is not read in 2 seconds, it is not read at all. Therefore, the "No-Read Timer" parameter in the bar code decoder is set to 2 seconds. Refer to the *DS/DD Series B Enhanced Bar Code Decoders (Bulletin 2755) User's Manual*, publication 2755-833, for details concerning the configuration of your Allen-Bradley Enhanced Bar Code Decoder.

Operating Sequence

With the bar code decoder configured as previously described, the following series of events take place when a product with a good bar code label breaks the photo switch and this input to the SLC goes from false-to-true. The SLC 5/03 ladder program logic to make it happen is also included. Please note that, as previously stated, a bar code/SLC system may be configured in a variety of ways.

Important: *Messages sent by the SLC processor to the Enhanced Bar Code Decoder must be programmed as shown by the example ladder program on page H–29. If this logic is not followed, the communication between the two DH-485 devices could become out of sequence, resulting in no data transfers between the decoder and the SLC processor. To correct such a problem, cycle power to the decoder.*

Sequence of Events



NOTE: These events are described in more detail by the comments listed within the example ladder program page H–29.

Optimizing MSG Time-Out

If the time delay between sending a command to an Enhanced Bar Code Decoder and “polling” for the reply is not long enough, the MSG instruction will time-out (MSG TO bit = 1) each time it is enabled from that point forward. To re-synchronize the SLC processor and the decoder, you need to cycle power on the decoder to clear its buffer.

There are other ways of clearing the buffers in the decoder, such as sending a “Clear Buffers” command or a “Reset” command to the decoder. However, the best way to handle this issue is to never let it happen. Optimizing the time delay between sending the initial command and “polling” for the reply is the best way to accomplish this. The delay must be long enough so the decoder has enough time to formulate a reply to the command and short enough to not impact the throughput of the application.

Example MSG Instruction Configuration

The example SLC 5/03 and SLC 5/04 ladder program demonstrates how to send commands to an Enhanced Bar Code Decoder, and then after a time delay, “poll” for a reply. The internal setup screen parameters for the two MSG instructions in the example ladder program are shown below, along with the necessary Enhanced Bar Code Decoder configuration parameters.

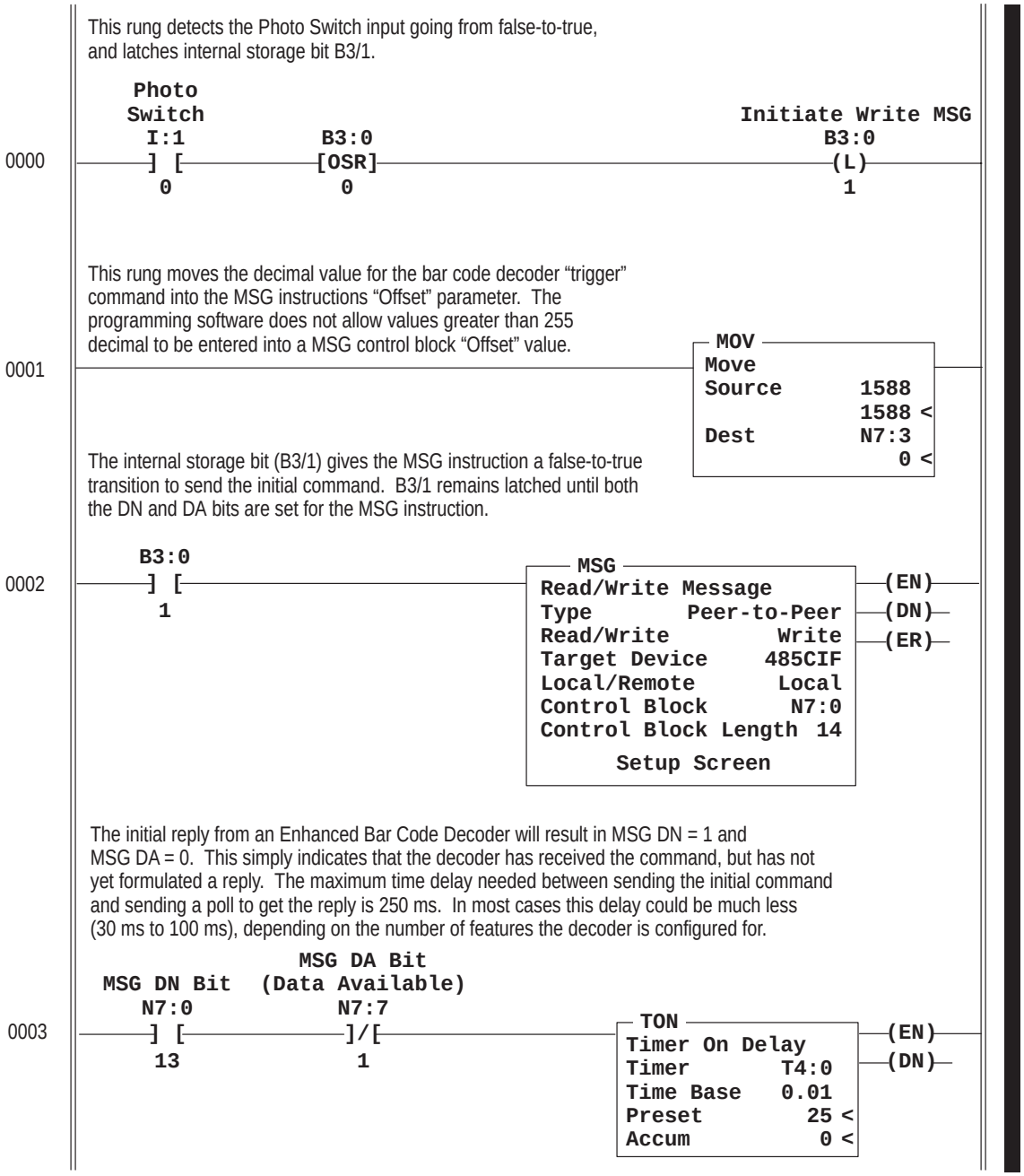
SLC 5/03 and SLC 5/04 Internal Set Up Screen Parameters		
	MSG #1	**MSG #2**
Type	peer-to-peer	peer-to-peer
Read/write	write	read
Target device	485CIF	485CIF
Local/remote	local	local
Control block	N7:0	N7:20
Channel	1	1
Target Node	2	2
Our source file address	N7:15	N7:40
Target CIF offset①	0	0
Message length in elements	1	10
Message time-out (seconds)	5	5

① The Target CIF Offset when working with Enhanced Bar Code Decoders as slaves on DH-485 must contain a value greater than 255. However, 255 is the largest value SLC programming software allows you to enter into this parameter in a MSG instruction. Therefore, use an unconditioned rung with a MOV instruction to move the proper value into the Target CIF Offset field. The example ladder program in this section demonstrates this. Note that 1588 decimal in a “MSG Write” is the value which results in a properly configured Enhanced Bar Code Decoder to initiate the “trigger” function. A value of 256 in a “MSG Read” requests a specified number of words of data from the bar code decoder. In this example, we are reading 10 words or 20 characters (bytes).

Example Scanner and Decoder Configuration

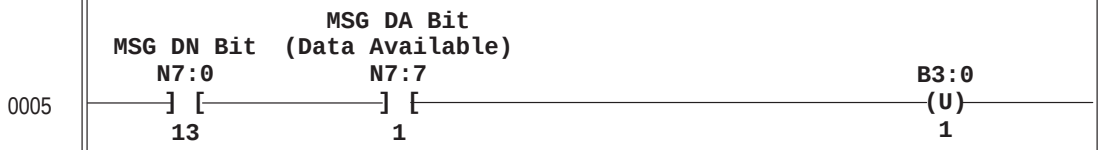
Scanner Configuration Parameters		2755-DS/DD Series B Enhanced Bar Code Decoder Configuration Parameters	
Scanner Control Page		**Host Communications Page**	
Discrete I/O:	Read Package 25ms No-Read Package 25ms	Baud Rate:	19200
Laser Light:	Triggered	Bits/Char:	8 Data 1 Stop
Decode Mode:	Host	Parity:	Even
No-Read Time:	2000 ms	Host Protocol:	DH485 PCCC-1
Inter Scan Time:	none	Device Address:	2
Capture Count:	2	ACK Char:	none
Symbols/Scan:	1	NAK Char:	none
Symbols/Package:	1	Large Buffer:	No
Match Complete:	1	Send Host Message: Package	Immediately after Valid
		Transmission Check:	none

Example Ladder Program



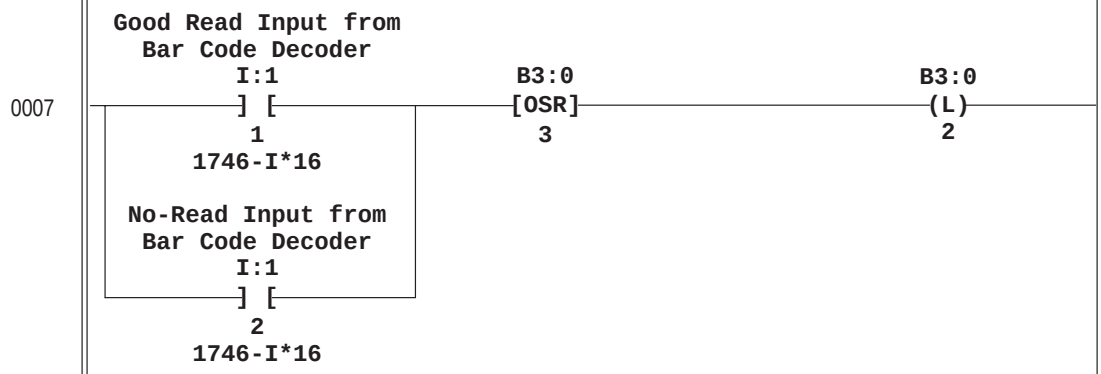
The internal storage bit, B3/1, holds the MSG instruction true until DN and DA are both set, indicating completion of the command sent and reply received sequence. When DN is set and DA is reset, unlatching the MSG EN bit effectively toggles the MSG instruction the same as if the MSG rung were toggled, i.e. rung conditions made false, then true. The MSG instruction is toggled one time after DN and NOT DA plus some time delay, to send a final poll to the decoder to get the MSG reply. When the reply is received, the SLC processor sets DN and DA.

When the SLC processor sets both DN and DA for a MSG instruction, the MSG sequence to an Enhanced Bar Code Decoder is complete. In this case, the decoder has received the "trigger" command and has performed this command. Therefore, unlatch B3/1 at this time to be ready for the next request for "trigger".



If an error occurs with the MSG instruction, the ER bit is set. If this occurs, the user can either try to resend the same message again by unlatching EN, or at this point, could sound an alarm or route the product down a rework loop or some other similar action. If the latter choice is used, you must also unlatch B3/1 at this time to be ready for the next request for "trigger".

This rung detects false-to-true transitions of either a good read or a no-read input from the bar code decoder and latches internal storage bit B3/2. B3/2 then, in the next rung, initializes the MSG read command to the decoder. This is done if either a good read or a no-read occurs, because the no-read message configured in the decoder is data as much as actual bar code label data. Therefore, the ladder program must distinguish between data that means no-read as well as actual bar code label data.



This rung moves the decimal value for the bar code decoder “Read” command into the MSG instruction’s “Offset” parameter. The programming software does not allow values greater than 255 decimal to be entered into a MSG control block “Offset” value.

0008

```
MOV
Move
Source      1588
            1588 <
Dest        N7:3
               0 <
```

The internal storage bit, B3/2, gives the MSG instruction a false-to-true transition to send the initial command. B3/2 remains latched until both DN and DA are set.

0009

```
B3:0
] [                MSG
 2                 Read/Write Message          (EN)
                   Type       Peer-to-Peer     (DN)
                   Read/Write      Read        (ER)
                   Target Device   485CIF
                   Local/Remote    Local
                   Control Block   N7:20
                   Control Block Length 14
                         Setup Screen
```

The initial reply from an Enhanced Bar Code Decoder will result in MSG DN = 1 and MSG DA = 0. This indicates that the decoder has received the command, but has not yet formulated a reply. The maximum time delay needed between sending the initial command and sending a poll to get the reply is 250 ms. In most cases this delay could be much less (30 ms to 100 ms), depending on the number of features the decoder is configured for.

0010

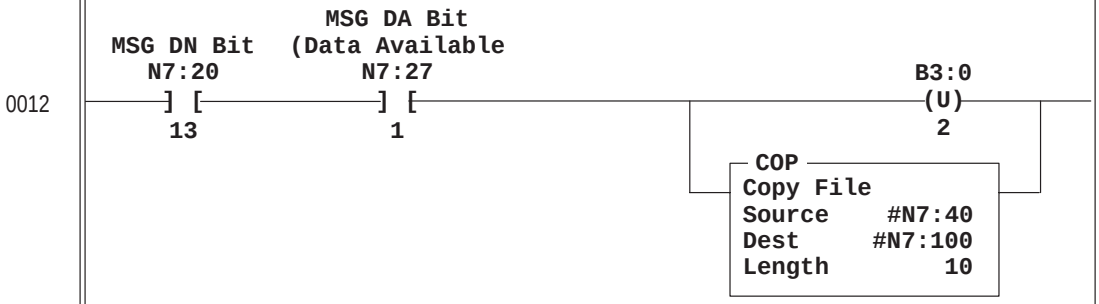
```
MSG DN Bit   MSG DA Bit
               (Data Available)
N7:20          N7:27
] [            ]/[                TON
13              1                 Timer On Delay      (EN)
                                  Timer      T4:1     (DN)
                                  Time Base  0.01
                                  Preset      250 <
                                  Accum         0 <
```

The internal storage bit, B3/2, holds the MSG instruction true until DN and DA are both set, indicating completion of the command sent and reply received sequence. When DN is set and DA is reset, unlatching the MSG EN bit effectively toggles the MSG instruction the same as if the MSG rung were toggled, i.e. rung conditions made false, then true. The MSG instruction is toggled one time after DN and NOT DA plus some time delay, to send a final poll to the decoder to get the MSG reply. When the reply is received, the SLC processor will set DN and DA.

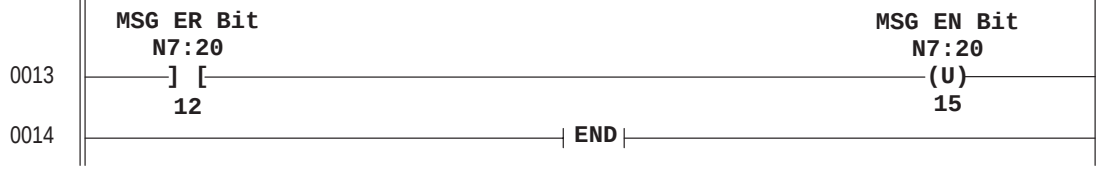
0011

```
T4:1                                          MSG EN Bit
] [                                             N7:20
DN                                              (U)
                                                15
```

When the SLC processor sets both DN and DA for a MSG instruction, the MSG sequence to an Enhanced Bar Code Decoder is complete. In this case, the decoder has received the "Read" command and has formulated a reply to this command. Therefore, unlatch B3/2 at this time to be ready for the next "Read" request. In addition, when DN and DA are both set, this indicates that the data received with the read reply (except "no-read" data) is valid and may be buffered or used.



If an error occurs with the MSG instruction, the ER bit is set. If this occurs, the user can either try to resend the same message again by unlatching EN, or at this point, could sound an alarm or route the product down a rework loop or some other similar action. If the latter choice is used, you must also unlatch B3/2 at this time to be ready for the next "Read" request.



Index

A

B

C

D

G

H

I

J

L

M

N

O

P

T

U

W

X

Z

Rockwell Automation
Allen-Bradley

Allen-Bradley, a Rockwell Automation Business, has been helping its customers improve productivity and quality for more than 90 years. We design, manufacture and support a broad range of automation products worldwide. They include logic processors, power and motion control devices, operator interfaces, sensors and a variety of software. Rockwell is one of the world's leading technology companies.

Worldwide representation.

Argentina • Australia • Austria • Bahrain • Belgium • Brazil • Bulgaria • Canada • Chile • China, PRC • Colombia • Costa Rica • Croatia • Cyprus • Czech Republic • Denmark • Ecuador • Egypt • El Salvador • Finland • France • Germany • Greece • Guatemala • Honduras • Hong Kong • Hungary • Iceland • India • Indonesia • Ireland • Israel • Italy • Jamaica • Japan • Jordan • Korea • Kuwait • Lebanon • Malaysia • Mexico • Netherlands • New Zealand • Norway • Pakistan • Peru • Philippines • Poland • Portugal • Puerto Rico • Qatar • Romania • Russia–CIS • Saudi Arabia • Singapore • Slovakia • Slovenia • South Africa, Republic • Spain • Sweden • Switzerland • Taiwan • Thailand • Turkey • United Arab Emirates • United Kingdom • United States • Uruguay • Venezuela • Yugoslavia

Allen-Bradley Headquarters, 1201 South Second Street, Milwaukee, WI 53204 USA, Tel: (1) 414 382-2000 Fax: (1) 414 382-4444

Publication 1747-6.15 – January 1998
Supersedes Publication 1747-6.15 – January 1996

PN 956500–02