WARNING

The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise should be allowed to program, install, alter, and apply this product. Potential bodily injury, death, or equipment damage could result if the product is improperly applied to any equipment application.

CAUTION

SY/MAX devices contain electronic components that are very susceptible to damage from electrostatic discharge. DO NOT handle this device by the gold edge contacts.

A static charge can accumulate on the surface of ordinary plastic wrapping or cushioning material. If any SY/MAX device must be returned to Square D, the following packaging instructions must be followed:

PREFERRED: Use the original packaging material as supplied by Square D. Place the device inside the metallized static-shielding bag.

ACCEPTABLE: Wrap the device in some type of antistatic material. Antistatic plastic material can be identified by its pink color, and can be obtained in sheet or bag form.

UNACCEPTABLE: Do not use ordinary plastic film, foam, or styrene chips ("popcorn" or "peanuts"). These materials can accumulate static charges in excess of 10,000 volts, resulting in possible damage to the SY/MAX electronic components.

CAUTION

Improper handling may cause permanent damage to this device.

1) **Never remove this device from the rack while power is ON. *Turn power supply switch to OFF and wait until all* indicating lights are off before removing.**
2) **Do not subject to static discharge.**
3) **Do not touch gold edge contacts.**

NOTICE

The products and services described in this manual are useful in a wide variety of different applications. Therefore, the user and others responsible for applying the products and services described herein are responsible for determining their acceptability for each application. While efforts have been made to provide accurate information within this manual, the Square D Company assumes no responsibility for the application, completeness or usefulness of the information contained herein.

UNDER NO CIRCUMSTANCE WILL THE SQUARE D COMPANY OR ANY OF ITS SUBSIDIARIES BE RESPONSIBLE OR LIABLE FOR ANY DAMAGES OR LOSSES, INCLUDING INDIRECT OR CONSEQUENTIAL DAMAGES OR LOSSES, ARISING FROM EITHER THE USE OF ANY INFORMATION CONTAINED WITHIN THIS MANUAL OR THE USE OF ANY PRODUCTS OR SERVICES REFERENCED HEREIN.

No patent liability is assumed by the Square D Company with respect to the use of any of the information, products, circuits , programming or services referenced herein.

The information contained in this manual is subject to change without notice.

Table of Contents

Addendum

Subject: SY/MAX®

Class 8055 Type SCP654, 655
Model 650 Processor
(Addendum for Instruction Bulletin 30598-730-01B1)

When using the IO/NET™ Level 1 Communication System with a Model 650 processor, a Revision 4.0 (or later) processor must be used. This addendum applies to Section 3.6.2, page 3-9 of the referenced Model 650 Instruction Bulletin. It is a clarification to the "NOTE" at the bottom of the second column on page 3-9.

> **NOTE:** When a Revision 4.0 (or later) processor is used with the Class 8030 Type CRM250 Local IO/NET™ Interface Module, up to 4096 registers may be transferred to the Local Interface for use as external registers.

1 MODEL 650 PROCESSOR OVERVIEW

1.1 Description

This bulletin describes the hardware and firmware features and installation procedures for the SY/MAX Type SCP-654 and SCP-655 processors.

The SY/MAX Model 650 is a powerful single-slot processor that features a built-in ThinWire Ethernet communication port. Coupled with SY/MAX support software running on a host computer, the Model 650 provides a flexible solution for high speed information transfer between the manufacturing floor and computer applications. The Model 650 has a powerful instruction set and can handle more than 8000 I/O in medium to large control applications. The Model 650 may be regarded as a control processor along with an equivalent "Ethernet NIM". Refer to Figure 1.1 for an example of how registers are divided between the Control Processor and the Ethernet NIM.

1.2 Ethernet Connectivity

The built-in ThinWire Ethernet port on the Model 650 complies with the IEEE 802.3a standard and provides high speed (10Mbit/second) Ethernet connectivity to one or more host computers and other Model 650 processors. Model 650 processors are networked using standard ThinWire components and can be connected to existing Ethernet networks. Model 650 processors can coexist on Ethernet with other non-SY/MAX devices and protocols such as DECnet and TCP/IP (Transmission Control Protocol/Internet Protocol). A host computer is not required for Ethernet communication between Model 650 processors.

Coupling the Model 650 processor with Class 8055 Type SFW390 communication support software for VAX/VMS provides direct Ethernet connectivity to any DEC VAX/VMS computer. A combination of up to 100 Model 650 controllers and VAX host computers running SFW390 Communication Software can exist on one network. SFW390 is easily linked to host application software using callable subroutines. The support software provides data register read/write, program upload/ download/verify, external forcing and a variety of alarm handling functions. Unsolicited messaging (report by exception) is also supported.

A Model 650 processor uses simple standard SY/MAX read/write instructions to communicate via the built-in Ethernet port (SY/MAX Channel 3) to other Model 650 processors or host computers. Each Model 650 processor has a switch selectable SY/MAX address that is inserted in the address field of the Ethernet packet. This allows standard SY/MAX routing procedures to be used for Ethernet network communication.

1.3 SY/MAX Compatibility

In addition to the high speed Ethernet port, the Model 650 has two RS-422 serial SY/MAX communication ports (Channels 1 and 2) which permit communication to other SY/MAX processors via a direct point-to-point connection or over the SY/NET® Local Area Network using a SY/NET Network Interface Module (NIM). The Model 650 can function as a data concentrator to other SY/MAX compatible processors, collecting data, and sending it via Ethernet to the host computer.

The Model 650 plugs into a register slot of any SY/MAX digital or register rack but external I/O can only be directly controlled from the CPU slot in the rack. Ladder programs from other SY/MAX controllers can be ported to the Model 650 requiring only minor modifications involving addition or modification of SY/MAX read/write instructions to facilitate Ethernet (Channel 3) communication.

1.4 Hardware Features

The Model 650 takes advantage of multiple microprocessors and co-processors for optimum performance. With a powerful scan processor, the Model 650 offers scan speed of less than 1 millisecond per K of ladder logic. In addition to the scan processor, a math co-processor accelerates math operations while a separate Motorola 68010 processor and an AMD 7992 Ethernet controller provide efficient Ethernet communications.

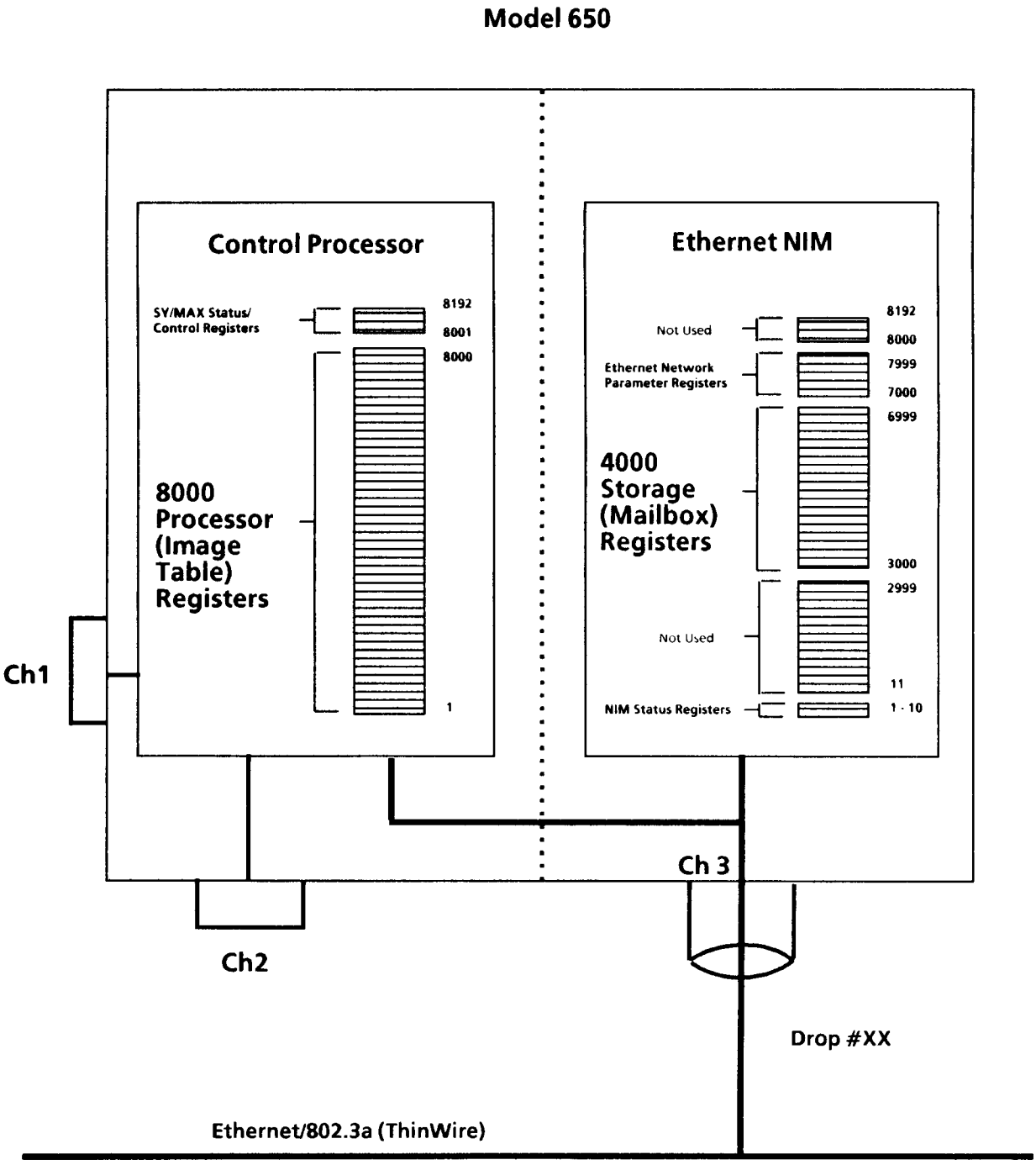


Figure 1.1 Model 650 Registers

The Model 650 is available in a 16K (SCP-654) or 26K RAM (SCP-655) memory size. Both versions have an additional 8K of processor image table registers on-board. Each processor register can be assigned as a data value, 16 digital I/O points, timer, counter, or other processor functions. In addition to the 8K of processor registers, another 4000 storage (mailbox) registers are available in the Ethernet NIM for use as buffer registers for applications requiring a data concentrator. The storage (mailbox) registers can be accessed without impacting the Model 650 processor scan time,via port 3.

The Model 650 offers a real-time clock/calendar that may be used for interval timing of task execution and scanning operations. The clock/calendar and processor memory are additionally supported by an onboard lithium battery that allows the clock accuracy and contents of memory to be maintained when power is removed from the Model 650. The charge levels for the onboard lithium battery and the batteries in the SY/MAX power supply are monitored, and a front panel "BATTERY LOW" LED indicates when either or both batteries require replacement.

To assist in troubleshooting, the Model 650 makes use of diagnostic LEDs. The LEDs annunciate processor status such as ETHERNET COMMUNICATION ERROR. In addition to the LEDs, status registers can also be monitored for troubleshooting purposes.

To maintain system security, the Model 650 has a four-position keyswitch that allows selection of four operating modes: DISABLE OUTPUTS, HALT, RUN and RUN/PROGRAM. Multi-level software security features can be used to prevent data and control program alteration, restrict I/O forcing, prevent viewing of some or all of the user program, restrict network access of information, and provide password security.

1.5 Instruction Set

A versatile and powerful instruction set allows the Model 650 to perform over 100 functions. The instruction set includes full program scan control including GOTO, subroutine, and timed interrupt capabilities. Also included in the instruction set is drum sequencer and ASCII input capability. The Model 650 can process ASCII data as input from devices such as weigh scales and bar code readers, in addition to supporting SY/MAX ASCII output capabilities for ports 1 and 2.

The Model 650 has more than 60 floating point and special math functions as well as PID instructions for performing closed loop control. The math processing power of the Model 650 allows the user to off-load the host computer from the chore of handling additional math operations. Refer to the instruction Set Table for the full list of instructions.

1.6 Technical Comparison

FEATURES

In the terms of SY/MAX processor features and functionality, the Model 650 is similar to the Model 400's product characteristics including the instruction set, method of program execution, real-time clock/calendar, onboard battery, keyswitch positioning, operating modes, etc. The major addition of the Model 650's features and functionality lies in the capability to communicate over a high-speed Ethernet network. Other feature enhancements include:

- Increased number of onboard data storage registers to 8000 (retentive).
- Increased memory size to 26Kwords in the SCP-655.
- The ability to install multiple Model 650s in one rack assembly, though only the processor residing in the CPU slot can directly control external I/O.

Noted differences between the Model 650 and the Model 400 include:

- Physical size is approximately 2.5 inches deeper than the Model 400.
- Increased current requirements to 5500mA @ 5 volts.
- Ethernet-related features such as the BNC connector, configuration switches, and Ethernet LED.
- Addition of Ethernet port 3.

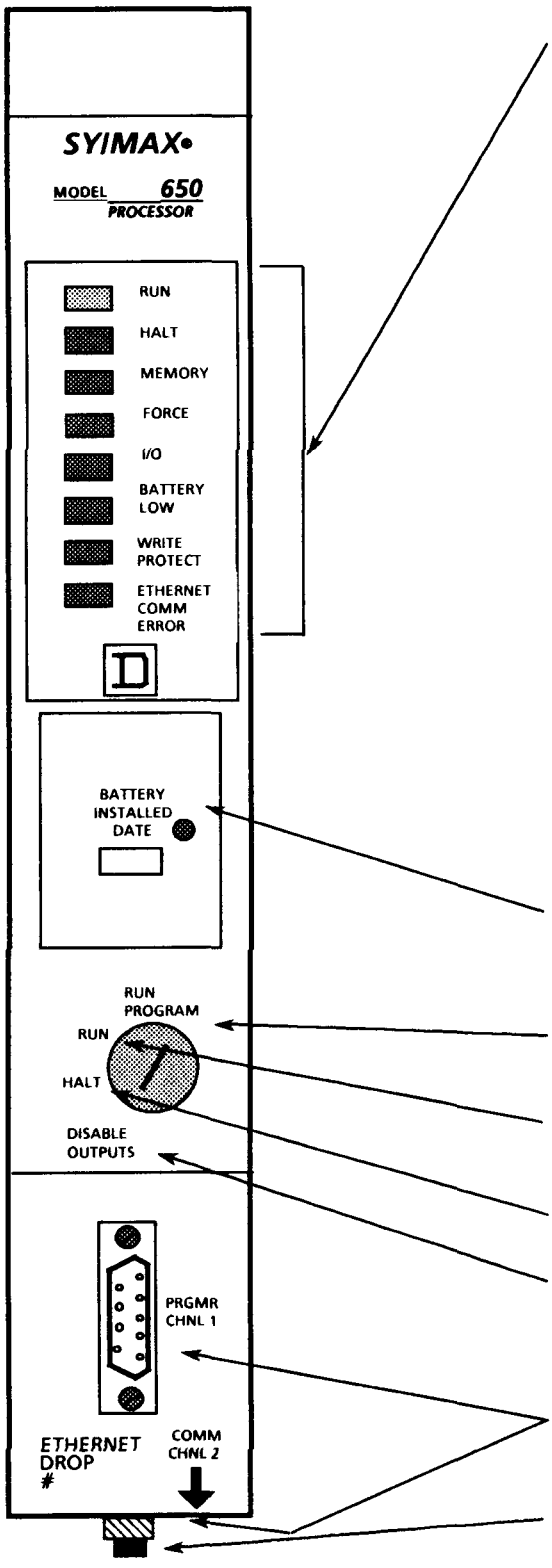


Figure 1.2 Model 650 Front Panel Identification

1.7 Front Panel Features

LED STATUS INDICATORS (See Section 4.2)

RUN (GREEN) - When ON steady, the processor is scanning ladder diagram program and operating on I/O's. When FLASHING, the processor is operating on ladder diagram program, but is not energizing any outputs (Disable Outputs mode).

HALT (RED) - When ON steady or FLASHING, the processor has halted its execution and is no longer scanning the program.

MEMORY (RED) - When ON steady in combination with the HALT LED FLASHING, indicates that the processor is halted due to a memory error.

FORCE (RED) - When ON steady or FLASHING, I/O have been forced to an ON or OFF state thereby overriding the ladder diagram program.

I/O (RED) - When ON steady in combination with the HALT LED FLASHING, indicates the processor has halted due to a malfunction in the I/O system. The I/O system is regarded as any I/O or register module that the processor controls in a rack assembly.

BATTERY LOW (RED) - When ON steady, indicates that the onboard battery is low. When FLASHING, indicates that the power supply batteries are low. The ON steady condition takes precedence.

WRITE PROTECT (RED) - When ON steady, indicates that the user memory is protected by the keyswitch positioned in the RUN mode and/or the internal write-protect jumper is in the NO PROGRAM position. When FLASHING, indicates some type of software-generated security is in effect. The ON steady condition takes precedence.

ETHERNET COMM ERROR (RED) - When ON steady, indicates a fatal "Ethernet" hardware error. When flashing, each off-on-off blink of the LED signifies one non-fatal network error. (The LED flashes for those Ethernet errors that affect this model 650 only.) Refer to Sections 4.4 and 15.8.

BATTERY ACCESS DOOR - Used to gain access to the processor's on-board backup battery. See Section 12.1.

KEYSWITCH (See Section 4.1)

RUN/PROGRAM - In this mode, the processor executes the ladder program and allows changes to be made to the program while in RUN.

RUN - In this mode, the processor executes the ladder program but does not allow changes to be made to the program while in RUN, or forcing to occur or be changed.

HALT - In this mode, the processor does NOT execute the ladder program and all outputs are turned OFF.

DISABLE OUTPUTS - In this mode, the processor executes the ladder program, but outputs are turned OFF.

COMMUNICATION PORTS (See Section 5)

RS-422 *serial differential ports for connecting programming and* peripheral equipment (COMM CHNL 2 on bottom front of unit).

ETHERNET port (CHNL 3, bottom rear) for connecting the Model 650 to a high-speed ThinWire Ethernet network.

2 SPECIFICATIONS

2.1 Model 650 Specifications

2.1.1 ELECTRICAL SPECIFICATIONS

Current Draw on SY/MAX Power Supply ..... 5500 mA ±10% @ 5VDC±5% The PS-10, 11, 40, or 41 power supplies must NOT be used, due to the current requirements of the Model 650.

Memory Support Time From SY/MAX Power Supply Alkaline D-Cell Batteries .... 4 month minimum (battery age less than two years with only one Model 650 in the rack). Primary backup power source for ladder program, storage memory, and clock/calendar when the Model 650 is in a rack during a power outage. Battery low voltage status can be monitored via contact status (bit 17 of register 8176). Current draw is 80 µA @ 4.2 VDC @ 40 °C.

Onboard Battery .. 3.5 to 3.6V AA lithium (Tadiran TL-5104 or SAFT LS6). Battery low voltage status can be monitored via contact status (bit 32 of register 8176).

Memory Support Time From Lithium Battery 9 month minimum (unloaded battery less than two years old). Secondary backup power source for ladder program, storage memory, and clock/calendar when the Model 650 is in a rack during a power outage. Primary power when Model 650 is removed from the rack. Battery may be replaced without powering down the Model 650. Current draw is 189 µA @ 3.6 VDC @ 40 °C.

Undervoltage Lockout Circuit ..... Halts and resets the Model 650 and removes it from the SY/MAX bus when incoming DC supply voltage falls below 4.6VDC.

2.1.2 ENVIRONMENTAL SPECIFICATIONS

Ambient Temperature

Operational ..... 32 to 140 °F (0 to 60 °C)

Storage ..... –40 to 176 °F (–40 to 80 °C)

Relative Humidity .. 5–95%, noncondensing

Vibration ..... Vibrated in three axes from 5 to 14 Hz at 0.2" (5 mm) displacement, and 14 to 200 Hz (two G acceleration maximum) for 90 minutes per X, Y and Z axis. Dwell at resonance frequencies for 10 minutes.

Electrical Noise ..... Passes the following tests for noise:

- NEMA Part ICS 2-230 and IEC 65A Part 5 Section 2.6.2.4. (Showering Arc).
- ANSI / IEEE C37.90a, IEEE 472 and IEC 65A Part 5 Section 2.6.2.5. (Surge Withstand Capability).
- Chattering Relay (Landis) Test.
- Current Spike (Reversing Motor) test.
- ESD Protection meets IEC 65A Part 5, Section 2.6.2.2.

2.1.3 PHYSICAL SPECIFICATIONS

Dimensions
(without key) ..... Width- 1.5" (3.8 cm)
(Figure 2.1) Height- 12.8" (32.5cm)
Depth- 9.5" (24.2cm)
(with key) ..... Key adds an additional .75 inch (1.9 cm) to the depth when installed.

Weight (Approx.) .. 5.3 pounds (2.4 kg)

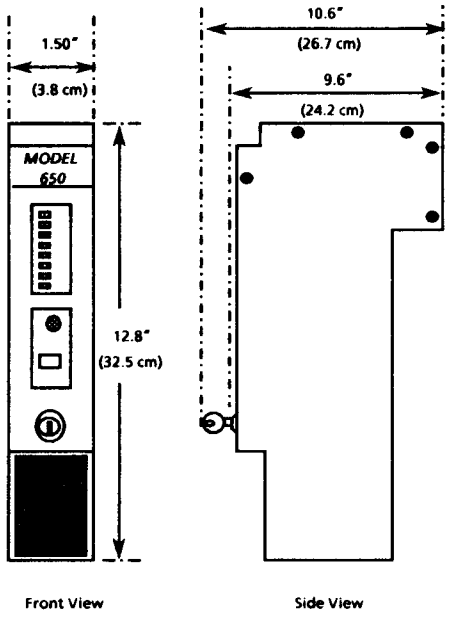


Figure 2.1 Model 650 Physical Dimensions

2.1.4 FUNCTIONAL SPECIFICATIONS

Scan Speed ..... 0.8 milliseconds per K
(SCP-654 and 655) Boolean
2.9 milliseconds per K nominal

NOTE: *I/O update from the image table for 128 local digital points adds a minimum of 0.3 milliseconds per scan. Refer to Section 14.2 for a discussion of execution times.*

Logic ..... Motorola® 68010 processor, Motorola 68881 math co-processor, and AMD 29116 for ladder scanning.

802.3/Ethernet and ThinWire Specifications

Type ..... IEEE 802.3 10BASE2 ThinWire Ethernet

Data Rate ..... 10 Mbits per second

Ethernet Communication . Motorola 68010 and AMD 7992 Ethernet Controller

Nodes Per Segment 30

Nodes Per Network 100 (using repeaters)

Maximum Segment Length ..... 607 ft. (185m)

Maximum End-to-End Network Length* ..... 3,034 ft. (925m or five 185m segments), using repeaters

Minimum Cable Length between Nodes ..... 20 in. (0.5m)

Maximum Drop Length ..... Less than 1 1/2 in (4cm), use direct connection to 'T' connector

Controller ..... AMD AM7990

Serial Link Interface ..... AMD AM7992

Coaxial Transceiver Interface ..... National Semiconductor DP8392

Real-Time Clock/Calendar .. Accuracy within one second per day @ 77°F (25°C), 16 seconds per day over full ranges:
temp. - 32 to 140°F (0–60°C)
relative humidity- 5–95%

* Applies to 10BASE2 ThinWire Ethernet networks. Longer distances are possible using repeaters with other media such as 10BASE5 Standard (ThickWire) Ethernet. See Appendix D.

Memory:

PROCESSOR TYPE	USER-MEMORY SIZE	MEMORY TYPE
SCP-654	16 Kwords	RAM
SCP-655	26 Kwords nominal. (Refer to Section 14.1)	RAM

User Memory Utilization ..... 1 word per contact.

User Memory Overhead ..... 8 words

Front Panel LEDs ... RUN, HALT, MEMORY, FORCE, I/O, BATTERY LOW, WRITE PROTECT, and ETHERNET COMM ERROR.

External I/O ..... Over 8,000 digital and over 3000 analog. Any mix in groups of 4, 8, 16, or 32 *digital points or* 4, 8, *or* 16 analog points per module.

Internal Relay Equivalents ..... 16 per unused register (8,000 nominal total).

Processor Registers ..... *Up to 8000 retentive 16-bit* integer registers or 4000 retentive 32-bit floating-point registers (in addition to user program memory)

User-defined Storage (Mailbox) Registers ..... 4000 retentive 16-bit integer registers (in addition to processor registers and user program memory.)

Register Module Compatibility ..... All SY/MAX modules except Class 8030 Type CRM-115 and CRM-116 Bus Expander/Terminator. (Refer to Section 3.1 for more information.)

2.2 New Features

The following lists enhancements to the Model 650 that are not offered in the SY/MAX Models 300, 500, and 700 processors. The Ethernet feature is also not available in the Model 400.

- **Ethernet Communications** (Section 15).
- **Multiple Model 650s In the Same Rack** (Section 3.7).
- **Real-time Clock/Calendar** (Section 12.2).
- **Automatic Storage of Error Codes** (Control Register 8107, Section 13.2)
- **ASCII Input** (Section 10.1).
- **Drum Sequencer** (Section 11).
- **Onboard Lithium Battery** (Section 12.1).
- **RUN and RUN/PROGRAM Modes** (Keyswitch Positions, Section 4.1).
- **Variable ROUTE Rungs** (Section 5.5).
- **Undervoltage Lockout Circuit** (Section 3.5).

2.3 Register Usage

The Model 650 can rack address up to 8000 user-definable registers, each having a unique address value of 1 to 8000. These registers are located in an image table in the Model 650 processor and are used to reflect external I/O and internal relay status, store data, build shift registers, and accumulate time or count data in timers and counters. Additional registers are used to indicate overall programmable controller system status such as errors and keyswitch positions.

Each register is composed of 16 data bits and 16 status bits as illustrated in Figure 2.2. The data bits are accessible to the user for data storage and may be represented in binary or word form. The status bits cannot be changed or altered directly by the user, but can be programmed as contacts or indirect status reads and monitored in the user memory.

Control registers 8001 to 8192 are used to control and monitor the basic operation of the Model 650. These registers are user-alterable, with exception of registers 8179 to 8192 which are read-only system status registers, and can only be monitored. See Section 13 for details on the control registers.

The floating point (FLP) mode combines two contiguous 16-bit data registers to form one 32-bit floating-point data register. FLP registers allow manipulation of numbers that contain floating decimal points. Refer to Section 7 for an explanation of "Floating Point Math".

Figure 2.3 illustrates the register usage for digital I/O points, analog I/O points, integer storage registers, and floating-point storage registers. Each digital I/O point occupies one bit of one register; 16 digital I/O points requires one complete register. Each analog I/O point occupies all 16 bits of one register and two complete registers are needed for each floating-point register.

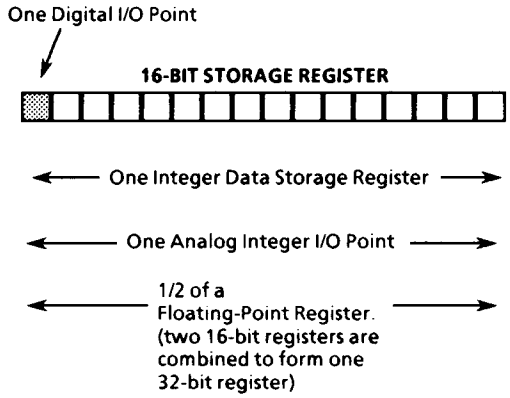


Figure 2.3 Register Use

The actual hardware I/O capacity of the Model 650 is over 8000 digital points and 3000 analog points. Typically, the maximum 26,000 word user memory is exceeded before the hardware maximum capacity is reached.

In addition, the Ethernet NIM has its own unique set of over 5,000 registers. Register addresses 1 through 10 contain NIM status information, 11 through 2999 are not accessible, 3000 through 6999 are retentive storage registers, and 7000 through 7999 contain Ethernet network parameters. Ethernet NIM registers 1-10 and registers 3000-7999 are battery-backed. Refer to Section 15.4 for additional information.

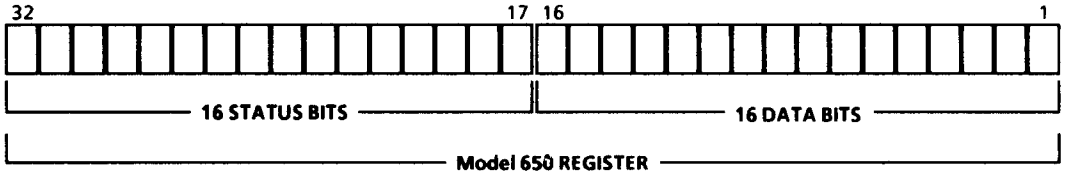


Figure 2.2 Model 650 Register Structure

2.4 Instruction Set for All Types

Figure 2.4 Model 650 Instruction Set

Instruction	Display	Description	Remarks
Contacts	⊣ ⊢ ⊣ / ⊢	Represents ON/OFF status of inputs, outputs, internal relays, and register bits. Normally OPEN (NO) or normally CLOSED (NC).	Input devices can be monitored in HALT.
Coil	—( )—	Represents an output connected to the processor.	Upon a transition to RUN, all coils are de-energized on the first scan. During the second scan, all coils are updated depending on the solution of the logic in that rung.
Internal Relays	R– —( )—	Represents control relay functions.	Operation is similar to an output, but output devices are not directly controlled.
Latch/Unlatch Relays	—(L)— —(U)—	Duplicates the action of a mechanically-held latching relay. This is accomplished through the use of two internal relay coils with the same address. Once a latch coil is energized, it remains ON until the unlatch coil is energized, even if power is removed.	It is recommended that latch and unlatch rungs be programmed ***sequentially***.
Transitional Coil	—(T)—	An internal or external coil which is only energized on a transition of the logic from OPEN to CLOSED. The output remains ON for a single processor scan. The logic must turn the coil OFF and transition to ON again to re-energize the output coil.	Requires two consecutive I/O addresses, one to record the transition and one to remember the last state. A transitional coil *MUST* be assigned an ODD address number. The processor uses the next I/O address to record the last state of this same coil.
Master Control Relay	MCR —( )—	MCR ON allows logic to function. MCR OFF turns the outputs except latched coils OFF. MCR controls all succeeding rungs until a new MCR coil is programmed or the end of the program is reached.	
Timer	TMR	Timer register stores a four digit decimal value from 0 to 9999. Selectable time bases are 0.01 sec., 0.1 sec., and 0.1 min.	Allows for programming ON/OFF delays and interval timers.
Counter	CTR	Counter register stores a four digit decimal value from 0 to 9999. UP, DOWN, and UP/DOWN counters are standard.	Counters can be cascaded to obtain a count greater than four digits.
Drum Sequencer	DRUM	Simulates the action of a rotary drum switch.	Manual or automatic time, event, or a combination of time/event sequencing.
Synchronous Shift Register	SHFT	Used to shift individual bits of a storage register in either a forward or reverse direction. 1, 8, or 16-bit channels can be shifted at a time.	Used in applications that require the user to keep track of bit patterns.
Asynchronous Shift Register (First In First Out)	FIFO	Data is loaded into the first zone and exits out the last zone using IN and OUT commands respectively. 8 or 16-bit channels can be shifted at a time.	Used to record the order of events as they occur, such as in conveyor or alarm annunciation.

Instruction	Display	Description	Remarks
Data Comparison	IF [=, ≠, ≥, <]	Compare functions between storage registers, between an integer and a constant, or between a floating-point storage register and a constant. When an IF comparison is true, the IF box acts like a short circuit.	Up to three comparisons can be programmed on one line. In *integer* mode, numbers from **–32,768** to **32,767** can be compared. In *floating point* mode, a range from **–3.4 x 10^{38}** to **3.4 x 10^{38}** and values as small as **± 1.2 x 10^{-38}** are allowed. Math and SPECIAL functions can be used in IF statements.
Data Transfer	LET	Transfers data from one storage register to another or presets a constant into a storage register.	Ranges are the same as Data Comparison (above). Math and SPECIAL functions can be used in LET statements.
Transitional LET, IF	T LET T IF	Transition-sensitive; restricts the operation of the LET or IF instruction to one operation for each OPEN to CLOSED transition of the input condition.	
Indirect Register Read	RDSTAT RDDATA FNDBIT	Indirect register read of *status* field. Indirect register read of *data* field. Locate and clear the first bit (lowest bit number) set in the indirect register being read.	
Binary to BCD	BCD	Conversion of binary data in a storage register to its binary coded decimal (BCD) equivalent within a LET or IF instruction.	Allows the processor to drive a BCD-type LED display.
BCD to Binary	BIN	Conversion of binary coded decimal (BCD) data to its binary equivalent, within a LET or IF instruction.	Used in applications to convert the ON/OFF thumbwheel switch position status to binary form for processor readability and usage.
Standard Math Operations (Integer and floating point math): **Addition** **Subtraction** **Multiplication** **Division** **Square Root** **Absolute Value** **Change Sign** **Sine** **Cosine** **LOG_{10}** **Ln_e** **$(Y)^x$**	LET, IF + – X ÷ SQRT ABS NEGATE SIN COS LOG LN Y**X	Math operations are performed from left to right, and are programmed into LET or IF instructions. Value range for *integer* math operations: **–32,768 to +32,767** Value range for *floating-point* math operations: **– 3.4 x 10^{38} to + 3.4 x 10^{38}**, not smaller than **± 1.2 x 10^{-38}**. See Section 7, *"Floating-Point Math"*.	The result of any *integer* math operation must not exceed **± 32,767**, or an overflow condition will result. The *intermediate* result of an *integer* computation cannot fall outside **± 2,147,483,647**, or an overflow condition will result. The result of any *floating point* math operation must not exceed **± 3.4 x 10^{38}** or be smaller than **± 1.2 x 10^{-38}**, or an overflow condition will occur. The *intermediate* result of a *floating point* computation cannot fall outside the range of: **± 3.4 x 10^{-4932} to ± 1.2 x 10^{+4932}**.

Instruction	Display	Description	Remarks
Advanced Math and Trig Operations	INVERT	Divide 1 by the current result (1/x).	
	ROUND	Round floating point number to integer.	
	XSQRD	Square the current result.	
	HYPOT	Calculate 2-dimensional hypotenuse.	
	HYPXYZ	Calculate 3-dimensional hypotenuse.	
	TANGNT	Calculate tangent.	
	COSEC	Calculate cosecant.	
	SECANT	Calculate secant.	
	COTAN	Calculate cotangent.	
	ARCSIN	Calculate arc sine.	
	ARCCOS	Calculate arc cosine.	
	ARCTAN	Calculate arc tangent.	
	ARCCSC	Calculate arc cosecant.	
	ARCSEC	Calculate arc secant.	
	ARCCOT	Calculate arc cotangent.	
	SETDEG	Perform subsequent calculations in degree mode.	
	SETRAD	Perform subsequent calculations in radians mode.	
	DEGRAD	Convert degrees to radians.	
	RADDEG	Convert radians to degrees.	
		The following constants are stored to 80-bit precision:	
	PI	Multiply result by pi (3.14159....).	
	L2T	Multiply result by log base 2 of 10.	
	L2E	Multiply result by log base 2 of natural log e.	
	LG2	Multiply result by log base 10 of 2.	
	LN2	Multiply result by log base e of 2.	
	EULER	Multiply result by Euler's Constant.	
	E	Multiply result by natural log e.	
	1DEG	Multiply result by 1 degree in radians.	
	1RAD	Multiply result by 1 radian in degrees.	
	ETOPI	Multiply result by e^{pi}.	
	PITOE	Multiply result by pi^{e}.	
	ETOE	Multiply result by e^{e}.	
	ETOEULR	Multiply result by e^{Euler}.	
	SQRTE	Multiply result by square root of e.	
	SQRTPI	Multiply result by square root of pi.	
Statistical Operations	ADDSTA	Store current result in statistical block of registers consisting of the summation of previous values, summation of the squares of previous values, and the accumulated count of previous values.	
	VARNCE	Calculate statistical variance on the accumulated totals in the statistical block.	

Instruction	Display	Description	Remarks
PID Computations	PIDR PIDD PIDM LEDLAG	Calculate reverse-acting PID loop Calculate direct-acting PID loop Calculate PID loop in manual Calculate lead/lag function for PID loop	
Accumulator Manipulations	YTOALT	Same as $(Y)^x$ except X in this case is taken from the alternate accumulator instead of the second argument.	These manipulations maintain an 80-bit precision when performing calculations that extend beyond a single rung.
	SWAP 16	Exchange low and high *bytes* of the current intermediate result	
	SWAP 32	Copies high *word* into low *word* of the current intermediate result	
	ACCALT	Exchange current accumulator with the alternate accumulator.	
	ADDALT	Add current intermediate result from the alternate accumulator.	
	SUBALT	Subtract current intermediate results from the alternate accumulator.	
	DUPALT	Copy alternate accumulator into current accumulator.	
	DUPACC	Copy current accumulator (intermediate result) into alternate accumulator.	
	HYPALT	Same as HYPOT except the alternate accumulator is used instead of the second argument.	
	ATANYX	Similar to ARCTAN except that the function is performed on the ratio Y/X.	
Identify Number Type	NUMTYP	Identifies mathematical classification of the current intermediate result.	
Forcing	*CRT INITIATED*	Forcing overrides the actual input status or output as controlled by the ladder diagram.	Refer to Section 14.4.
Serial Communication to Other Processors or Peripherals	T READ T WRITE T ALRM T PRNT	Copy storage register and I/O status between devices which may be located up to 10,000 feet apart. All four are transition-sensitive.	Requires no additional interface hardware.
Read	T READ	Copies storage register data from remote device to local processor.	
Write	T WRTE	Copies storage register data from a local processor to a remote device.	
Alarm	T ALRM	Sends a numerical value from a local processor to a remote device	Numeric values from **-32,768** to **+32,767** can be transferred.

Instruction	Display	Description	Remarks
Print	T PRNT	Sends ASCII-coded messages from the processor to any device that communicates in ASCII. See Section 13.2.	Simplified means of producing alarm messages, production reports, etc.
ASCII Input*	T PRNT	Accommodates a block size of up to 256 characters with block termination based on either character count or a delimiter character. Supports various word structures including parity. See Section 10.	Allows the Model 650 to interface directly with bar code readers and weigh scales, etc. * ASCII Input command not applicable for ETHERNET (Channel 3) port.
AND	∧	A logical function used within an IF or LET command with the following property: if **X** and **Y** are two logic variables, then the function "X AND Y" is defined as: **X** **Y** **X AND Y** 0 0 0 0 1 0 1 0 0 1 1 1	Integer mode only.
OR	∨	A logical function used within an IF or LET command with the following property: if **P** and **R** are two logic variables, then the function "P OR R" is defined as: **P** **R** **P OR R** 0 0 0 0 1 1 1 0 1 1 1 1	Integer mode only.
Exclusive OR (XOR)	⊕	A logical function used within an IF or LET command with the following property: if **S** and **T** are two logic variables, then the function "S XOR T" is defined as: **S** **T** **S XOR T** 0 0 0 0 1 1 1 0 1 1 1 0	Integer mode only.
Random Number Generator	RANDOM	Generates a random integer number	Results in a positive random number that ranges from zero to the value of the argument minus 1.
GOTO	GOTO	Instruction to cause the processor to skip over a section of the program to another specified position in the program. Position is determined by a MARK rung placed in memory.	Shortens processor scan time during the immediate jump to a new rung position in the program. Can jump forward or backward.

Instruction	Display	Description	Remarks
Subroutines	GOSUB	Group of ladder diagram rungs that can be executed from the user's ladder diagram. Only enabled by a GOSUB instruction programmed in the main program.	Applications include variable sequence machines, recipe programs, and redundant processes.
	MARK ST SUB	Denotes the beginning of the subroutine area.	
	RTN	The end of a subroutine. Causes the processor to jump out of the subroutine and return to the main program.	
Timed Interrupt, Communications and Register Security, Scan Time Limit	S 8165	A special safeguard rung using register 8165 allowing the following features: 1. Communications Inhibit 2. Program Viewing Inhibit 3. Register Safeguarding 4. Timed-Interrupt Scan Control 5. Scan Time Limit (See Section 6.8 for details.)	Protecting programs. Used in conjunction with: **8176** **—( )—** **-16**
Matrix	M LET M IF M TIF M	A matrix is a group of storage registers handled as a unit. Formulates multi-register data transfers and math functions. Also allows multi-register compare statements.	Applications include machine diagnostics and menu selection, etc.
Array	A LET A IF A	An array is a group of 32-bit floating point data storage registers operated on as a unit. Formulates multi-register data transfers, compares, and math functions	
End of Scan Communications Update	S 8170 —( )— -16	If set to 1, the processor finishes servicing the incoming communications message before resuming the ladder scan. See Section 5.6.1.	
Immediate I/O Update	S 8176 —( )— -07	Setting bit 7 of register 8176 to "1" (via a ladder rung) causes the Model 650 to interrupt its memory scan and immediately update selected external I/O points. Inputs are written *to* the image table and outputs are read *from* the image table.	Registers to be updated are user-defined.
Bus Write to Microcell	BWRITE	Allows the processor to immediately update Microcell storage registers via the backplane.	Refer to Section 8.6.
Immediate Communications Update	S 8176 —( )— -11, -12 or-13	Processes incoming messages from communication ports 1, 2, or 3 received during processor scanning.	
Extended EOS Comm Servicing	S 8170 —( )— -16	Allows the processor to utilize more than 5 msec at the end of each ladder scan to finish processing the entire block of registers associated with an incoming COMM message.	Helps preserve data integrity for multiple register blocks; refer to Section 5.6.1.

3 INSTALLATION

This section defines the following topics for the Model 650 processor:

- Differences in Operating Characteristics Compared to Other SY/MAX Devices and Certain Functions
- Rack Configurations
- Use of Register Modules
- Installation Considerations
- Undervoltage Lockout Circuit Operation
- Power Installation Troubleshooting
- Rack Addressing the Model 650 Programmable Controller System

3.1 Differences In Operating Characteristics Compared to Other SY/MAX Devices and Certain Functions

The Model 650 is compatible with other SY/MAX devices; however, performance and/or functional characteristics in relationship to the SY/MAX Model 300, 500, and 700 processors may be different. Note that replacing another processor in an existing system with the Model 650 may require additional changes and the system may not operate exactly as before. Due to the identical processor architecture and operating system of the Model 400, aside from the exceptions of Section 1.2, the Model 650 operates in a system the same as a Model 400.

For example, the Model 650's user ladder program is executed based upon the state of an onboard "image table". The image table is updated at the end of every scan. This process is similar to the Model 300 and 400 operation but is different from the method used by Model 500 and 700 processors. The 500 and 700 use real-time bus updates; register information being used to execute the ladder program resides in modules outside of the processor and not in an internal image table.

In some cases, the programmable controller system *may respond differently because of the Model 650's* increased processor speed. Also, differences in how the Model 650 implements certain functions, such as forcing, produces results that are not the same as observed in a Model 500 or 700 system.

CAUTION

If the Model 650 is replacing a SY/MAX *Model 300, 500, or 700 processor in an* existing system, READ the following SECTIONS to determine if certain processes must be changed before installing the Model 650.

The following modules produce characteristic differences when used with the Model 650:

1. **Type CRM-115/116 Bus Expander/Terminator Modules.** Due to bus timing delays, these modules are NOT compatible with the Model 650. In most cases, the Type EQ5138-G1/G2 Parallel Digital Driver/Receiver (PDD/PDR) modules can be substituted. Refer to Appendix C.3.

2. **Local Interface (LI) Modules.** These modules are compatible with the Model 650, but cannot be *used to provide additional storage registers for* the Model 650. The rack addressing that currently exists for another processor can be used with a Model 650 system. However, any registers that are assigned to an LI in the CPU rack and are *not assigned to remote drops* will adversely affect throughput. Refer to Section 3.6.5.

3. **Local Transfer Interface (LTI) Modules.** These modules are compatible; however, due to the image table in the Model 650, certain application considerations must be observed. Refer to Appendix B for operating considerations when using the Model 650 in a redundant LTI-equipped system.

4. **Parallel Digital Driver/Receiver (PDD/PDR) Modules.** These modules (Types EQ5138-G1/G2) are compatible with the Model 650 as long as certain rack addressing procedures are observed. Refer to the "Rack Addressing Register Allocation" discussion in Section 3.6.4.

   NOTE: *If a Model 650 replaces a Model 500 or 700 in an existing system that contains PDD/PDR modules, rack addressing must be reviewed to determine if it is suitable or if alterations are required.*

The following functions produce different results when used with the Model 650 in comparison to SY/MAX processors other than the Models 400 and 600.

1. **Results of Forcing I/O.** Forcing in the Model 650 is implemented outside of the image table and is actually performed directly on the external I/O. It is important to note that the forced state may not be reflected in the image table. For example, the contacts of a forced output reflect the ladder logic state and not the forced state of the output. Refer to Section 14.4 for more information.

2. **Timed Interrupt Subroutine.** Due to the end-of-scan (EOS) update operation of the image table, a timed interrupt which executes more than once per ladder scan operates on unchanged external I/O information. The timed interrupt tolerance must also now allow for the time required to complete an EOS update since an interrupt cannot be called while external I/O are being serviced. Refer to Section 6.8.1 for more information.

3. **Matrix IF Instructions.** The Model 650 may not scan some conditional elements in a rung that has no effect on the logical solution of the output. This optimizes scan time and increases processor scan performance.

   For example, if a rung contains a leading contact that is open, other conditional elements between the contact and the output element have no effect on its logical solution. If a rung contains a matrix IF instruction preceded by FALSE conditional logic, the matrix IF instruction is simply skipped over and not scanned. This differs from the Model 500 and Model 700 where bits 17 and 18 of the status register are reset when a matrix IF rung is executed FALSE. Thus, bits 17 and 18 in the status register of the matrix IF box should be ignored when using a Model 650 processor.

4. **Matrix Transitional IF Instruction.** As explained previously, the Model 650 does not always scan all conditional elements. The matrix TIF instruction executes differently than for the Model 500 and 700, and because of this the matrix TIF instruction must be preceded by a transitional contact.

   NOTE: *In order that the Model 650 behave as described in the programming manual, the Matrix TIF instruction must be preceded by a transitional contact (i.e., a contact that is energized by a transitional coil).*

5. ***SCP-344 RAM/UVPROM Processor.*** The SCP-344 requires the programming of two safeguard rungs that reference TLET S8164 and S8165. If a Model 650 replaces an SCP-344, these TLET rungs must either be removed or replaced with a TLET S8165 safeguard rung with new security features that are executed in accordance with Section 6.8.

6. **NUMTYP Instruction.** If the Model 650 replaces a Model 700 processor, the NUMTYP instruction yields slightly different results due to the difference in math coprocessors used. Refer to Section 8.6 for further details.

7. **Asynchronous Shift Register (FIFO).** If the Model 650 replaces a Model 300 processor, the FIFO operation is different in the handling of the last zone of the stack. In the Model 300, the DATA OUT zone is the last zone of the stack. In the Model 650 (as in the Model 500 and 700), the second-to-last zone is the bottom of the stack and the last zone is used as the DATA OUT zone. Because of this, the Model 650 does not accept a FIFO with a defined stack of less than three registers (ERROR 05 is generated). Refer to the programming Instruction Bulletin for more information.

8. **Memory Use.** If a Model 650 replaces any SY/MAX processor other than Models 400 or 600, more memory is typically required to store the same program in the Model 650. For example, each new rung requires an additional 1/3rd of a word, while each new parallel branch now consumes a complete word. A good rule of thumb when transferring a program to a Model 650 is to allow 10% more memory than the size of the original program. See Section 14.1 for more details on memory usage.

9. **Counter decode equal to 0.** If the Model 650 replaces a Model 300 processor which has a program containing a counter rung with a decode of 0, the Model 650 will de-energize its associated output whenever the logic in the CLEAR line is false; under the same conditions, the Model 300 will energize the output. Note both processors will energize the output when the decode value equals 0 and the logic in the CLEAR line is true.

10. **Editing in RUN/PROGRAM Mode.** If a Model 650 replaces a Model 300, 500, or 700 processor, the scan time may increase substantially for a single scan when an edit is performed while the processor is running. The difference in scan speed is a one-time occurrence and slower system response may be observed than when editing the same program in a Model 300, 500, or 700 processor. Refer to Section 14.2.2 for additional information when editing in the RUN/PROGRAM mode.

11. **Multiple Model 650s in the same rack.** While multiple Model 650s may reside in the same rack, only the processor in slot 1 can directly control external register modules and I/O. Also, communication via the backplane is not supported. Refer to Section 3.7 for rack addressing multiple Model 650s in the same rack.

12. **Special note on storing labels in a Model 650.** When using a programming package which predates the SFW-374 Universal Programming Package, there may be a situation when labels cannot be saved to a processor, or when labels saved to a processor may appear to occupy a large amount of memory. This only occurs when saving labels *to* a Model 650 processor.

    Labels can always be saved to disk with any SYM-xxx packages. The deluxe CRT programmer (SPR-300) will save labels to tape and to the processor correctly.

    The SFW-374 programming package allows for full label storage in a Model 650 processor.

3.2 Rack Configuration

Single or multiple Model 650(s) may be installed in any CPU or register slot of any SY/MAX register or digital rack assembly. The CPU and register slots are identified in each rack as shown in Figure 3.1. The CPU and register slots contain a rear edge connector that provides the appropriate bus lines, +5VDC, and common for the Model 650 operation. To directly control external I/O and the various I/O modules configured in any rack, the Model 650 MUST be installed in the CPU slot or slot 1.

NOTE: *If multiple Model 650s are installed in the same rack, only the processor in slot 1 can communicate over the backplane with the other modules. The only way a Model 650 residing in a slot other than slot 1 can communicate with other devices is through the Ethernet or serial ports. Refer to Section 3.7 for rack addressing a Model 650 not installed in the CPU slot (slot 1).*

If the Model 650 is installed in a single rack configuration and only four-function or eight-function digital I/O are used, rack addressing is not required. The I/O addresses automatically revert to the default for the local I/O. If the Model 650 is installed in a multiple rack configuration and register modules (including 16 or 32-function digital I/O) are used, the corresponding registers and locations must be rack addressed. See Section 3.6 for details on rack addressing.

3.3 Register Modules

Register modules (also referred to as intelligent I/O) usually contain an onboard microprocessor and can handle more complex input and output data manipulation than the digital I/O modules. Register modules include analog I/O's, BCD multiplexed I/O's, and high-speed counter modules.

The register modules are installed in register slots in the various racks. Depending upon the type of digital or register rack used, the number of register slots vary from 0 to 15. The following table and Figure 3.1 identifies the register slots in each type of digital or register rack:

CLASS 8030 TYPE	# OF USABLE REGISTER SLOTS
CRK-100 (Digital Rack)	NONE
CRK-210, CRK-300, DRK-210, DRK 300, GRK-110, GRK-210, HRK-100, HRK-200 (Digital Racks)	1 slot (R2)
HRK-150 (Digital Rack)	3 slots (R2 through R4)
RRK-100, RRK-200, RRK-300 (Register Racks)	4, 8, or 15 slots (R2 and above)

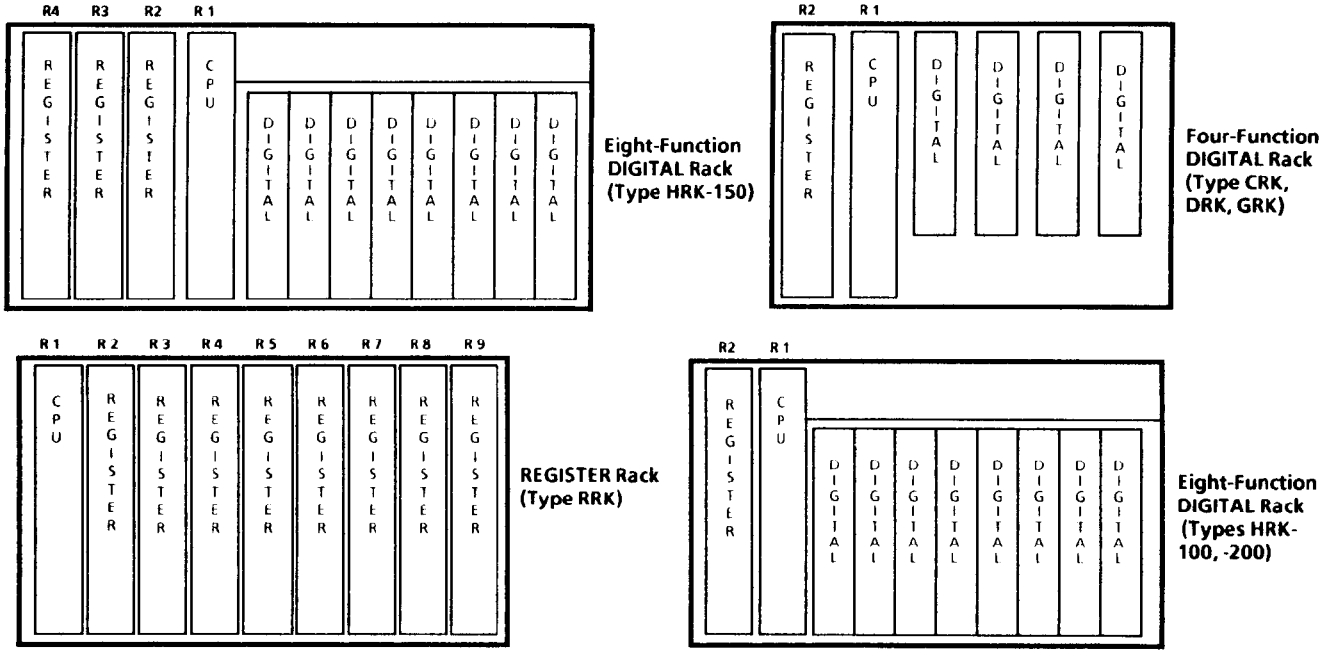


Figure 3.1 CPU and Registers Slot Locations in a Digital or Register Rack Assembly

3.4 Model 650 Installation

The following procedure describes the Model 650's physical installation into a SY/MAX digital or register rack.

1. Open the battery access door on the front of the Model 650 and observe the battery contact polarity as shown on the inside of the battery compartment cover. Install the lithium battery by matching the + and - polarities of the battery and the contacts in the battery compartment. Close the battery door and tighten the latching screw. Record the date of the battery installation on the the door.

2. If the Model 650 processor is to be installed in a CRK, DRK, or GRK four-function digital I/O rack, the side plate shipped with the Model 650 must be removed and replaced with an optional side plate P/N C30609-350-01A. (Contact Square D). The optional plate allows sufficient clearance between the side plate of the Model 650 and the edge connector in the rack.

3. The Model 650 requires +5VDC, common, and a proper current rating of 5500 mA to operate. Make sure that the SY/MAX power supply is capable of delivering the current required by all the devices in the rack and any other racks being driven by the same supply. The number of Model 650s in the same rack is limited to the possible current output of the power supply used; a single PS-31 or PS-61 should not be used to power more than 3 Model 650s.

NOTE: *A PS-10, 11, 40, or 41 does NOT supply enough current for a Model 650 system. Refer to Instruction Bulletin 30598-159-xx to calculate the total current requirement for each module and processor configured in the system and the proper power supply to use.*

WARNING

Do NOT remove or install a SY/MAX power supply or power cable to the rack with power applied. Turn OFF the power at the power supply and remove or unplug the incoming AC voltage line. Electrical shock, bodily injury, or damage to the equipment may occur if the power is not removed from the power supply before installation.

4. Make sure that the rack is mounted in an area where the temperature around the Model 650 never exceeds 140° F (60°C). Mount the rack so the Model 650 is installed in a vertical position.

5. Use a Square D Class 8030 Type CC-10 power cable to connect the Model 650's rack to the SY/MAX power supply. Connect the CC-10 cable from the power connector on the rack to the P1 connector on the power supply. If the CC-10 cable is not used, the BATTERY LOW LED on the Model 650 will falsely indicate a low battery.

6. *The DIP switches at the rear of the Model 650* have been set at the factory to all 1's, indicating non-Ethernet mode. The rotary switch factory setting will vary, but its setting will not affect the drop number, in this mode. Refer to Section 15 for Ethernet mode DIP switch settings before installing.

7. To directly control external I/O, the Model 650 MUST be installed in the CPU slot in the rack assembly. For installation into the CPU or any register slot, apply steadily increasing pressure to the front of the processor. Press until the Model 650 is firmly seated in the edge connector and is against the stud located above the CPU slot.

NOTE: *The Model 650 can ONLY directly control external I/O if installed in slot 1 (the CPU slot) of any digital or register rack. Multiple Model 650 processors can be installed in other register slots, however external I/O control is only provided via the CPU slot (1).*

9. After the processor is properly seated in slot R1 or a register slot, close the latching bar located above the module slots (not applicable for four-function racks).

10. Tighten the captive screw to the bottom of the processor for a solid connection and ground between the processor and the edge connector in any slot.

11. Install the AC power line to the SY/MAX power supply and turn on the power switch on the SY/MAX power supply.

12. Multiple Model 650s in the same rack MUST communicate either via Ethernet or the serial communications ports. Backplane communications for multiple processors in the same rack IS NOT supported.

13. Refer to Section 15 for additional information on Ethernet hardware and installation considerations.

3.5 Undervoltage Lockout Circuit Operation and AC Fail Function

The Model 650 is designed with an onboard DC Undervoltage Lockout Circuit (ULC) that monitors the incoming DC voltage level at the edge connector in either the CPU or register slots. If the incoming DC voltage falls below 4.6 volts, the processor enters the HALT state and disconnects from the backplane bus. If a DC undervoltage condition occurs, the processor is not warned ahead of time to execute an orderly shutdown.

If multiple Model 650s are installed in the same rack, due to circuit tolerance differences and power distribution characteristics it is possible that some processors might continue operating while others have shut down if the DC power is marginal.

When the incoming DC voltage rises above 4.6 volts, the Model 650 executes a standard power-up initialization sequence, but remains in the HALT mode. To resume the RUN state, the keyswitch must be toggled from RUN to HALT then back to RUN. When the ULC is triggered, an ERROR 902 (Section A.3.7) is posted in register 8175, identifying the failure.

NOTE: *When the ULC system is activated, none of the LEDs are illuminated on the Model 650.*

Since the ULC only resides in the Model 600, 650, and Model 400, other devices located in the same rack may remain powered up. Modules under processor control behave as though the system is in HALT.

NOTE: *Non-bus modules, such as the D-LOG and Network Interface modules, may remain operating since their power requirements are less stringent than the Model 650's.*

AC FAIL FUNCTION

The Model 650 also monitors an incoming DC signal identified as AC FAIL. AC FAIL is generated from all SY/MAX power supplies and indicates when the incoming line side voltage to the power supply has been lost or degraded.

The operation of the AC FAIL is different from the ULC because the AC FAIL signal alerts the processor to execute an orderly shutdown. The ULC is not able to alert the processor of an impending DC power loss. Abnormal conditions, such as an overloaded power supply or cable disconnection between the rack and the power supply, can cause an unexpected interrupt of DC voltage.

As long as the Model 650 is receiving adequate power, at least one of the eight LEDs on the processor's front panel will be ON. If all of the Model 650 LEDs are OFF (or flashing erratically), a problem resides in the power source and should be repaired or replaced. Use the following procedures in Section 3.5.1 to determine the various power problems that may be the cause of a malfunctioning Model 650.

3.5.1 TROUBLESHOOTING POWER PROBLEMS

If the system is powered up and none of the LEDs are illuminated on the Model 650, use the following procedure to determine the problem:

1. Make sure the SY/MAX power supply is receiving proper incoming AC voltage.

2. Check the inline AC fuse and replace if blown.

WARNING

Remove incoming AC power to the power supply before checking or replacing the AC inline fuse. Electrical shock or bodily injury could occur.

3. Check that the power supply is able to deliver the proper amount of current for the system configuration. Refer to Instruction Bulletin 30598-156-xx or 30598-159-xx for a listing of current draws for various SY/MAX modules.

NOTE: *Each Model 650 requires 5500mA @ 5VDC to operate.*

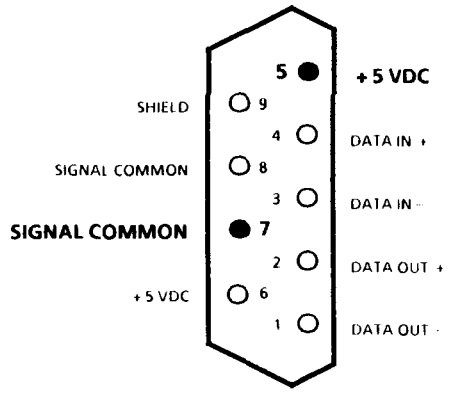


Figure 3.2 COMM and PRGMR Port Voltage Test Points

If the problem persists:

Check for +4.75 to +5.25VDC across pins 5 and 7 of the Model 650's serial port (COMM or PRGMR port) with a volt meter. Use Figure 3.2 for pin and signal identification.

- If the voltage across pins 5 and 7 is zero, use the following to determine the problem:

Power is not reaching the rack:

1. Check that the CC-10 cable between the power supply and the rack is securely connected and is making good contact.

2. Visually check that all the pins in the connectors in the power supply, the rack, and on both ends of the cable are the same height. If one or more are lower, pull the pin out until it is the same height as the other pins.

3. Check the continuity of the wires in the CC-10 cable with a meter. Replace the cable if an OPEN is detected.

4. Check the voltage at the P1 connector on the SY/MAX power supply. If power is not present at the P1 connector, replace the power supply.

DC power is being short-circuited:

1. Turn OFF power to the rack and check that all of the modules are properly seated. Turn ON power and see if the problem is resolved.

2. If step 1 did not resolve the problem, turn OFF power to the rack then remove and reinstall each module one at a time. Turn ON power to the rack after each module is reinstalled and check for proper operation.

3. If a voltage problem is found in a slot of the rack or the rack has been damaged, replace the rack.

- If the voltage across pins 5 and 7 is greater than zero but less than 4.6 VDC, the ULC has properly prevented operation of the Model 650. Use the following procedure to identify why the low input voltage is being supplied from the power supply.

  1. As previously mentioned, the problem could reside in the integrity of the CC-10 cable or cable connections. Refer to steps 1 and 2 in the section **"Power is not reaching the rack"** to check the CC-10 cable.

  2. Check for excessive current draw from other modules in the rack. Check the current requirements for each module in the system and the power supply current rating (see Instruction Bulletin 30598-159-xx). If the power supply is rated under the total system current requirement, replace the power supply with an appropriately rated power supply.

  3. Once these possibilities have been eliminated, replace the power supply.

If the previous troubleshooting procedures failed to resolve the problem, simplify the system by turning OFF the power to the rack and remove all of the modules from the CPU rack except for the Model 650. Turn ON the power to the system and see if the Model 650 powers up properly.

Power OFF the system, reinstall the next module, and turn power ON checking for proper operation. Continue with this procedure for the modules that are to be reinstalled until the problem module or slot is identified. Make sure that the power to the rack is turned OFF before removing or installing ANY module. This procedure offers the best systematic approach for isolating the power or component failure as quickly as possible.

3.6 Rack Addressing a Model 650 in the CPU Slot

A Model 650 Programmable Controller system consists of a Model 650 processor, digital and/or register racks with SY/MAX power supplies, and various interface and function modules. A great amount of flexibility exists when configuring and addressing Model 650 system components. Because of this flexibility, proper rack addressing is extremely important.

NOTE: *The following sections ONLY apply to a Model 650 installed in slot 1, the CPU slot. For rack addressing a Model 650 that is NOT installed in slot 1, no registers are assigned to any slots. Refer to Section 3.7.*

3.6.1 INITIAL SYSTEM LAYOUT

Rack addressing is determined by the physical layout of a programmable controller system and is based on the following procedure:

1. Determine the physical location of the CPU rack and all other desired digital I/O and register racks.

   NOTE: *Remote racks can be located a total cable length of 10,000 feet (3050 meters) away from the CPU rack if LI/RI modules are used. (Refer to LI/RI Instruction Bulletin 30598-247-03 for more information.) PDD/PDR modules limit the distance between the racks to 6 feet.*

2. At each drop location, list the digital I/O requirements (type and quantity) along with any needed register modules. This determines the total number of modules, racks and power supplies needed.

3. Based on the total number of I/O modules, determine the number of Local Interface (LI) modules and Remote Interface (RI) modules needed. See Instruction Bulletin 30598-247-XX, "SY/MAX Local/Remote Interface" for further details.

3.6.2 PROCEDURE

The Model 650 executes the user's ladder logic program based on an image table. This image table, which can be visualized as being composed of external I/O and data storage registers, is updated (inputs are read from, and outputs are written to, the external world) at the end of every scan. An immediate I/O instruction can be user-programmed to momentarily interrupt a scan to update a user-defined portion of the image table (refer to Section 13.2 for information on bit 7 of register 8176).

The Model 650 has 8000 on-board registers; in this respect the Model 650 differs significantly from the SY/MAX Model 500 or 700. The Model 500 has 460 on-board registers but is able to address up to 2008; the Model 700, with no on-board registers, can address up to 8000. These "extra" registers used by Models 500 and 700 do not actually exist until other register modules such as the Local Interface are installed into the CPU rack.

The Model 650 is able to address all 8000 of its on-board registers. This means that it is no longer necessary to add Local Interface modules to obtain extra storage registers, since these extra registers already exist within the Model 650. Hence, the only purpose for adding a Local Interface module to the Model 650's CPU rack is for the serial communication link to any remote I/O.

Local Interface modules can be installed in slot R2 of the four-function and eight-function digital racks, and in any slot (except slot R1) of the HRK-150 and register racks.

NOTE: *In the RRK-300 rack, slots 17 and 18 cannot be used for a Local Interface.*

Sufficient registers should be assigned to each Local Interface module to account for all present and anticipated future external I/O, both digital and register, that will exist on remote drops.

NOTE: *A single Local Interface module can service up to 255 external I/O registers.*

Refer to Section 14.2 for information on what effects rack addressing and register allocation have on system response, and Instruction Bulletin 30598-247-XX for more information on Local and Remote Interface modules.

3.6.3 DETERMINING RACK ADDRESSING NEEDS

Use the following procedure to determine addressing needs:

1. **Create a system layout sketch** showing the CPU rack, and all other racks with I/O wiring. This layout sketch ensures proper assignment of racks to Local Interface channels. Each Local Interface module has two channels (1 and 2); each channel can address 8 racks (drops).

2. **Address the programmable controller system.** Install the Model 650 processor in the R1 CPU slot in either a digital or register rack. The rack addressing (RK ADDR) mode of the CRT programmer or SFW Programming Software can be used to allocate registers in the system as discussed below.

3. **Install other modules into the register slots of the CPU rack.** Addresses for the modules that are installed in the register slots should be assigned in ascending order. If the CPU rack is a digital rack, assign the CPU slot sufficient registers to handle the local digital I/O.

4. **Assign sufficient registers** to the Local Interface (LI) module to satisfy requirements for external remote I/O based on the content and quantity of remote drops.

5. **Assign registers to other register modules in the CPU rack** such as analog I/O's, BCD multiplexed I/O's and high speed counter modules in accordance with their Instruction Bulletins.

6. **Address remote digital I/O racks and remote register racks** within the range of registers previously assigned to the Local Interface module(s) during the CPU rack addressing. Local Interface modules communicate with these racks via a Remote Interface module located in each remote rack.

   Only those addresses assigned to the Local Interface slot in the CPU rack can be utilized by the remote racks. Assign registers to all Channel 1 drops sequentially, followed by all Channel 2 drops. There is no need to assign registers to the Local Interface modules to act as storage registers, since the Model 650 contains all needed storage registers.

   6A. Addressing the remote digital I/O racks. For all CRK, DRK, GRK, and HRK racks, slot 1 is always assigned as many registers as necessary to handle the local digital I/O. The Remote Interface module is always inserted into slot 1 of the remote digital I/O rack. Slot 2 is a register slot and should be addressed in accordance with the register module residing in that slot.

   6B. Addressing the remote register racks. Register rack slots are numbered from left to right; slot 1 is the leftmost. The Remote Interface module resides in slot 1. In the case of the register racks, the Remote Interface is not addressed. Register modules are inserted into the slots to the right of the Remote Interface. The number of registers assigned to that slot is dependent on the individual register module requirement.

7. **Spare address registers,** for future expansion, can be assigned to any unused slot number for any remote drop. The slot that is assigned with the spare addresses does not even have to physically exist. For example, addresses can be assigned to slot 2 (register slot) of a CRK-100 (a rack without a register slot). Any register not assigned to real world I/O is data-retentive.

NOTE: *Any addresses assigned to remote drops utilize serial I/O update times as shown in the Local/Remote Interface Instruction Bulletin. These update times must be taken into consideration when calculating the throughput of a system.*

8. **Data storage registers** exist as a continuous "block" of registers. This block begins with the last address assigned to the CPU rack and ends with register 8000.

   NOTE: *Unlike the Model 500 and 700 processors, data storage registers do not have to be assigned to a slot to physically exist. All 8000 registers exist in the Model 650, they are ALWAYS available for use – even if not assigned via rack addressing. For example, if the last register assigned to a slot in the CPU rack is register 200, then registers 201 to 8000 are available for use as internal I/O or storage registers, despite the fact they haven't been assigned as such.*

3.6.4 RACK ADDRESSING REGISTER ALLOCATION

- Default rack addressing (8000 registers in CPU slot 1) applies when the Model 650 is installed in a digital I/O rack with no companion register modules and no remote I/O.

- If the Model 650 is installed in a digital I/O rack in a system containing register modules and/or remote I/O, assign enough registers to slot 1 for the local digital I/O.

- If the Model 650 is installed in a register rack, registers do not need to be assigned to slot 1.

- Assign only necessary external I/O registers to Local Interface and other register modules in the CPU rack; include room for future expansion. There is no need to assign data storage registers to a Local Interface; the Model 650's 8000 register count is fixed and this practice can adversely affect throughput.

- When assigning registers to a slot containing a Parallel Digital Driver module, some special application considerations are in order due to the image table operation of the Model 650. Observe the following rules:

  1. If expansion is to a single remote rack, the Parallel Digital Receiver MUST be connected to channel 1 of the Parallel Digital Driver. The CPU slot containing the driver should ONLY be assigned as many registers as required to accommodate the digital I/O in the remote rack.

  2. If expansion is to two remote racks, the rack connected to channel 1 of the Parallel Digital Driver MUST be an HRK-200 (the second rack can be any digital rack). The CPU slot containing the driver should be assigned as many registers as required to accommodate the digital I/O in the two racks. Thus, the first eight registers assigned will correspond to the I/O in the HRK-200 connected to channel 1, while the balance of the registers (up to a maximum of 16 if both remote racks are HRK-200s) will correspond to the I/O in the second rack.

NOTE: *All registers assigned to the Parallel Digital Driver must have corresponding EXISTING external I/O. Failure to do this (for example, an HRK-100 rack connected to a channel receiving more than four registers) could result in oscillating outputs in the rack.*

Refer to Appendix C.2.3 for additional information on PDD/PDR updates.

- When assigning registers to digital I/O in a register module installed in the CPU rack, assign one register to a 16-point module or two registers to a 32-point module. Register modules should have registers assigned to the slot that they reside in.

- For remote **digital I/O racks**, registers for the digital I/O addresses should be assigned to the slot occupied by the Remote Interface module.

- For remote **register I/O racks**, do not assign registers to the Remote Interface. Instead, assign them directly to the slots in which the I/O modules are installed, including register modules that contain digital I/O.

- To improve throughput when laying out a Programmable Controller system, avoid "fragmenting" registers, i.e., ensure that all 16 bits associated with a register are either all inputs or all outputs. See Section 14.2.3.

- Any external I/O point can be forced ON or OFF. See Section 14.4.

- Model 650 system availability of 8000 registers is fixed; adding Local Interface or other register modules will not increase the available registers.

- Any of the 8000 on-board registers in the Model 650 that are not assigned to a slot are available as storage registers.

For more information on rack addressing, refer to Appendix C and the programming device's Instruction Bulletin.

3.6.5 RACK ADDRESSING COMPATIBILITY WITH OTHER SY/MAX PROCESSORS

The Model 650 can use programming and rack addressing configurations designed for other SY/MAX processors. An example of how this is accomplished is shown in Figure 3.3.

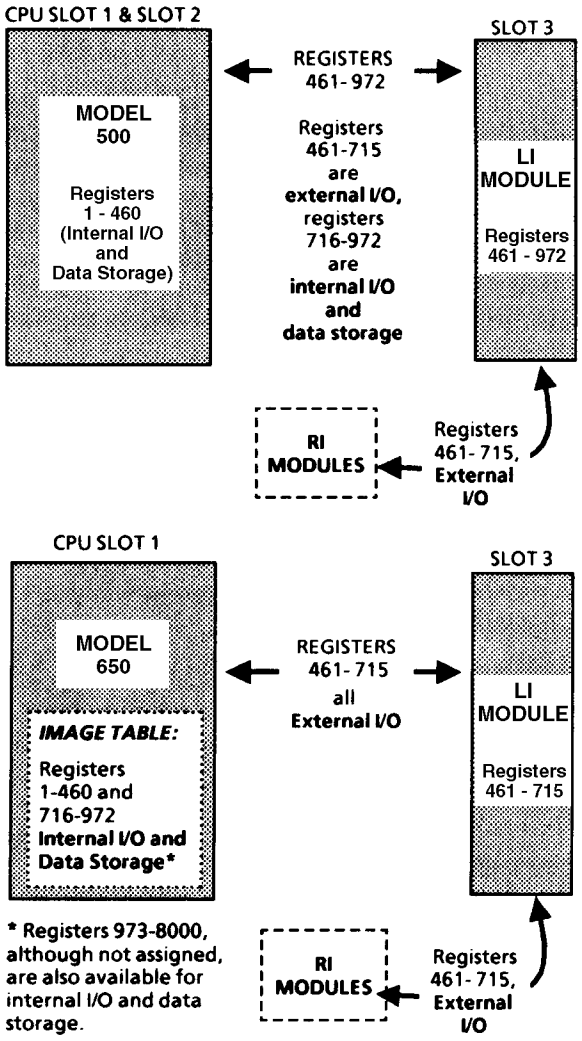


Figure 3.3 Rack Addressing, Model 650 vs 500

Refer to Figure 3.3. A common Model 500 configuration is to assign the first 460 registers to slot 1, and registers 461 through 972 to a Local Interface in slot 3. *External* I/O registers in this configuration begin at register 461 and could extend up to register 715 (although consuming all 255 external registers, the maximum for a single Local Interface, would be unusual).

Internal relays and storage registers in this example are located in the first 460 registers, and in registers 716 through 972. Note that for internal relays and data storage, the Model 500 (or 700) will access registers that physically reside in the Local Interface module. The Model 650 uses its own on-board registers for this purpose, however. The only purpose for the Local Interface when used with a Model 650 is to provide communication between Remote Interfaces and the CPU rack.

Because of the image table, *only registers that are exchanged with a Remote Interface need to be updated by the Model 650 in the Local Interface.* It would be a waste of processing time and a large amount of unneeded bus activity to shuffle storage registers between the Model 650's image table and the Local Interface.

In the case of the "old" Model 500 addressing, the Model 650 is designed to accept the register allocation as defined, and will determine which registers *need to be exchanged with the Local* Interface. Since the Model 650 "knows" that the Local Interface cannot transfer more than 255 registers to the Remote Interface, it will transfer only registers 461 through 715 to the Local Interface.

This 255-register transfer occurs because the user has addressed 512 registers to the Local Interface. The same 255-register transfer would have occurred had the user addressed 2008 registers to the Local Interface (as could occur with a type CRM-211).

The important thing to keep in mind is that this Model 650 "decision-making" process is transparent to the user, and allows existing rack addressing to be used "as is".

When a programmable controller system is being newly configured for a Model 650, there is no need to follow old rack addressing conventions for assigning registers to Local Interface modules.

System throughput will be adversely affected if up to 255 registers are unnecessarily assigned to Local Interface modules located in the CPU rack. For best throughput, *only registers assigned to Remote Interfaces should be assigned to the Local Interface.*

The example Model 650 configuration pictured in Figure 3.3 shows the absolute maximum number of registers (255) being transferred between the *processor and the Local Interface module.* The throughput of this example could be improved substantially if only the registers associated with actual drops are transferred between the Model 650 and Local Interface, as shown in Figure 3.3A.

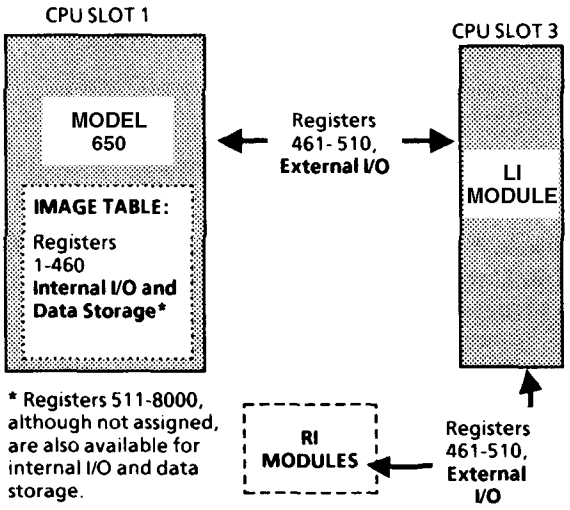


Figure 3.3A Model 650 Rack Addressed with 50 Remote Registers

If only 50 remote registers need be remotely addressed, the best configuration of the Model 650 shown in Figure 3.3 that would NOT require the re-addressing of registers in the program, appears as shown in Figure 3.3A. This is accomplished by assigning 460 registers to CPU slot 1, and 50 registers (461-510) to CPU slot 3. Registers above 510 would not need to be assigned, but would still behave as storage registers.

Refer to Appendix C.2.2 for additional information on the Local Interface module.

Rack addressing options when using existing programs in the Model 650 are therefore reduced to two:

1. **Preserve the existing rack addressing** for ease of transport between different processors, at a slight cost of throughput.

OR

2. **Modify the existing rack addressing** and register allocation to improve system performance.

CAUTION

If additional modules exist to the right of the Local Interface (LI) module being *addressed by the Model 650, the addresses* of these modules will change when the number of registers being assigned to the LI is changed. The Model 650's program must be altered to reflect the new addresses.

Refer to Appendix C.3 for a detailed discussion on the Model 650's compatibility with existing processor rack addressing configurations.

The system will operate properly with empty slots in the CPU rack, or the CPU rack can be altered to "close up" the empty slots. For example, since slot 2 is left empty when addressing a Model 500 or 700 system, the Local Interface could be placed in slot 2. An adjustment could then be made when rack addressing a Model 500-to-650 conversion to move all of the register assignments one slot to the left. This would free up an additional CPU rack slot, and eliminate the somewhat awkward condition of having slot 2 empty.

NOTE: *This procedure of moving the Local Interface to slot 2 would not require reallocating these registers – they would maintain their relative positions. It would, however, require adjustments to rack addressing and also that all modules in the CPU rack be moved one slot to the left.*

Refer to Appendix C for supplementary rack addressing information.

3.7 Rack Addressing Multiple Model 650s in the Same Rack

While multiple Model 650s may be installed in a rack for coprocessing applications, only the processor installed in the CPU slot (slot 1) can directly control digital and register modules via the backplane. Any coprocessing Model 650s installed in other than slot 1 will only draw power from the rack and cannot communicate via the backplane.

Rack addressing a Model 650 in slot 1 follows the same rules and procedures described in section 3.6. Performing a rack addressing "DELETE/CLR ALL" procedure on a processor in slot 1 will result in the default rack addressing of 8000 registers being assigned to slot 1. If additional Model 650s are installed in the same rack, and if the Model 650 in slot 1 has registers assigned to the slots in which they reside, these registers will behave as though they are assigned to an empty slot (i.e., as retentive data storage registers). Since these additional Model 650s only draw power, they are not active on the bus and any communication with them must occur either via Ethernet or the two serial ports.

Any processor NOT installed in slot 1 requires NULL rack addressing (i.e., no registers assigned to any slots) to be loaded or else it will not run (ERROR 985). NULL rack addressing can be created by performing a DEL/CLR ALL procedure on a Model 650 when it is NOT installed in slot 1. Following this procedure, no registers will appear assigned anywhere. The ladder program (with no entry for rack addressing) can then be loaded into the processor. Once developed, processor memory (NULL rack addressing and ladder rungs) can be recorded on tape or disk. Though not actually rack addressed, the first 8000 registers do exist internally, while the shared registers also operate as expected.

In summary, multiple Model 650s installed in the same rack obey the following rules:

- If installed in Slot 1, the Model 650 must be loaded with valid rack addressing. Do not assign registers to any slots in which additional Model 650s reside.

- If installed in other than slot 1, the Model 650 must be loaded with NULL rack addressing i.e., no registers assigned to any slots. NULL rack addressing is generated by performing a DEL/CLR ALL on the processor while installed in other than slot 1.

This distinction is essentially a safety measure which prevents processors from being accidentally interchanged; a processor intended for slot 1 operation will not run unless it is installed in slot 1, while a coprocessor (non-slot 1 processor) does not run unless it is installed in other than slot 1.

4 CONTROLS AND INDICATORS

4.1 Keyswitch Positions

The key-operated selector switch (Figure 4.1) locks the Model 650 into any one of four operating modes, each of which is described in this section. The key can be removed in any position.

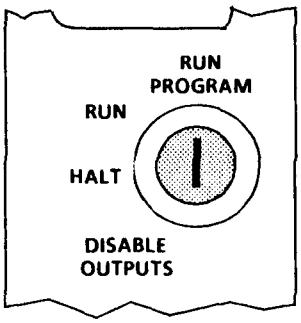


Figure 4.1 Model 650 Keyswitch

• RUN/PROGRAM

In this mode the processor functions the same as in RUN (below), but the program *can* be altered by the user via use of the override security position on the programmer. Forcing *can* be implemented and altered in this mode.

• RUN

In this mode the processor scans and executes the ladder program, and all external outputs are under control of the ladder program. While in this mode, the processor's program *cannot* be altered by any means. Forcing *cannot* be implemented in this mode; if already in effect, it cannot be altered.

IMPORTANT

When keyswitching between RUN and RUN/PROGRAM, do so as quickly as possible. An excessively slow transition (greater than one second in Rev. 2.0 and later, or one-fourth second in earlier revisions) may cause the Model 650 to go into HALT momentarily, then reinitialize before resuming RUN status.

• HALT

In this mode the processor is not scanning the program. All external outputs are turned off unless other special options are utilized.

• DISABLE OUTPUTS

In this mode the processor scans and executes the ladder program, but all external outputs are kept OFF. *Internal outputs and storage registers* (counters, timers, etc.) operate according to the program, as do the LEDs on I/O modules. This mode allows a program to be tested without actually energizing any external devices, thus eliminating the possibility of machine or process damage should the program be flawed in some way.

4.2 Indicator LEDs

As shown in Figure 4.2, the Model 650 has eight LEDs located on the front panel. Each LED, except where noted, has three possible states: ON, OFF, or FLASHING. The following describes the various states of the LED indicators.

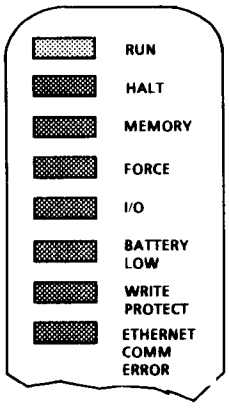


Figure 4.2 Model 650 Indicator Lights

CAUTION

If no LEDs illuminate or LED operation is erratic when power is applied to the Model 650, refer to Section 3.5.

As long as the Model 650 is receiving proper power from the SY/MAX power supply, *at least one* LED will be ON or FLASHING. If *all* LEDs (except RUN) come ON and stay ON, the Model 650 has failed to initialize properly. Check the processor-to-rack seating. An "all-ON" condition can also be caused by improper power supplied to the Model 650.

- **RUN**

ON: The processor is scanning the user program and is operating normally. This LED will light when the keyswitch is in either the RUN or RUN/PROGRAM position.

OFF: The RUN LED is off if the processor is halted by the keyswitch being in the HALT position, or if the processor is instructed to halt via the programming equipment or user program. This LED will also be OFF if the processor halts due to any Programmable Controller system malfunction.

FLASHING:The RUN LED flashes if the processor is in the DISABLE OUTPUTS mode due to one of the following conditions:

- Processor keyswitch is in the DISABLE OUTPUTS position.
- Processor has been instructed to run in the DISABLE OUTPUTS mode by the programming equipment.
- Processor has been instructed to run in the DISABLE OUTPUTS mode by the user program.

- **HALT**

ON: If the HALT LED is ON *continuously*, the Model 650's microprocessor is not operating.

Numerous diagnostic checks (power-up, halt-to-run, and run-time) are performed within the processor to ensure its proper operation. If an internal malfunction occurs, the processor halts, all external outputs are turned OFF, and the HALT LED illuminates. Any other non-internal failure will cause the HALT LED to *flash*.

OFF: The processor is operating in the RUN, RUN/PROGRAM or DISABLE OUTPUTS mode (depending on whether the RUN LED is ON continuously or flashing).

FLASHING: If the HALT LED is flashing, one of the following conditions has occurred:

- The key selector switch has been placed in the HALT position.
- The user program or programming device has instructed the processor to halt. In this case, the light will continue to flash until the HALT bit is cleared.
- A memory or I/O error has occurred in which case either the MEMORY or I/O light will also be ON.

If needed, the processor can be forced into the HALT mode by the ladder diagram program or by the correct sequence of keystrokes on a programming device by setting bits 1 or 3 of control register 8176 within the processor. These bits are called the HALT bit and the HALT/RUN bit respectively. For detailed information on these bits, refer to Section 13.

• MEMORY

ON: The processor has detected an error in the user memory. This condition can be caused by trying to run the processor with no ladder program or rack addressing in memory, or if a memory fault is detected in the ladder program. When the MEMORY LED is ON, the HALT LED will also be flashing.

OFF: User memory is operational.

FLASHING: Not used.

• FORCE

ON: Indicates that one or more inputs or outputs have been forced to an ON or OFF state, and that forcing via the COMM port (channel 2) is *disabled* (control register 8176 bit 5 = 1). Forcing overrides the actual input status or output commands of the ladder program. This forcing may have been via either the programming equipment or communication system. See Section 5 for information on the communication ports.

OFF: No I/Os are being forced.

FLASHING: Indicates that one or more inputs or outputs have been forced to an ON or OFF state, and forcing is *enabled* for the COMM port (control register 8176, bit 5 = 0).

• I/O

ON: A malfunction has occurred within the processor's internal I/O registers, or within the external input/output system. When the I/O LED is ON, the HALT LED will also be flashing.

OFF: Internal and external I/O systems are operational.

FLASHING: Not used.

• BATTERY LOW

ON: The lithium backup battery inside the Model 650 is low, and needs to be replaced. See Section 12.1 for battery information.

OFF: Both the onboard lithium battery *and* the power supply backup batteries are OK.

FLASHING: The power supply batteries are low, and must be replaced. If *both* the lithium battery and the power supply batteries are low, the steady ON condition prevails.

This condition also occurs if any cable other than a Square D Type CC-10 cable is used to supply the Model 650 with power.

• WRITE PROTECT

ON: All or part of user memory is protected from alteration. For this LED to be ON, either the keyswitch is in the RUN position or the processor's internal write-protect jumper is in the NO PROGRAM position. See Section 6 for information on Memory Security features.

OFF: No security locks are active.

FLASHING:
Some type of *software* security is in effect. See Section 6.

• ETHERNET ERROR

ON: Indicates that the Ethernet NIM communication processor is not running and cannot be communicated with. All communications attempted out of Ethernet port 3 will fail (a link error will be returned in the status register).

When this LED is on, register 8175 will contain error code 929. This error code refers the user to look at register 8094 for the specific cause of the failure. The Ethernet error codes stored in register 8094 are listed in Table 15.11.

OFF: Indicates that the Ethernet NIM is operating properly or is in non-Ethernet mode.

FLASHING:
A flashing LED signifies that non-fatal Ehternet errors are occurring. These errors (displayed in register 8094) are for informational purposes, and unless they occur repeatedly, do not require any remedial action by the user. During this time, the Ethernet NIM communication processor is still running and can be communicated with. The non-fatal error codes stored in register 8094 are listed in Table 15.12.

5 SY/MAX COMMUNICATIONS

5.1 Description

This section is intended as an overview of communication ports. As shown in Figure 5.1, the Model 650 has three communications ports (channels). PRGMR port is port 1, COMM port is port 2 and the Ethernet port is port 3. Two 9-pin RS-422 serial differential ports are for SY/MAX protocol and ASCII communications. The third, a BNC-type Ethernet port, connects to a high-speed Ethernet communication network. This section applies *primarily to the RS-422 ports; refer to Section 15 for* detailed Ethernet information.

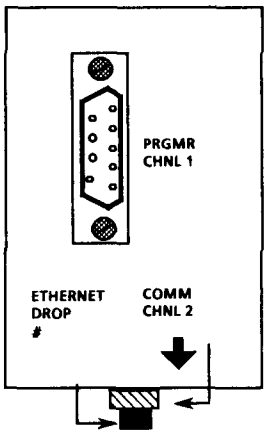


Figure 5.1 Model 650 Communications Ports

The PRGMR and COMM ports work identically except that the PRGMR port allows *complete* access to the Model 650's systems while the COMM port can be set up to lock out certain functions. See Sections 6.8 to 6.13 for inhibit functions for all ports.

All three ports support block READ/WRITE commands for a maximum count of 128 registers. All *ports also have an automatic timeout feature that* monitors communication integrity when using READ, WRITE, or ALARM statements. Ports 1 and 2 support PRINT statements for both ASCII input and output. However, port 3 supports ASCII output, but not ASCII input. Furthermore, although port 3 supports ASCII output, currently no devices are available on the Ethernet network to receive ASCII.

Refer to Section 10 for additional ASCII communication information.

Immediate Communications Update is supported for ports 1, 2 and 3. Refer to Section 5.6.1 for more information on Immediate Communications Update.

The pinout for both RS-422 communications ports is shown in Figure 5.2.

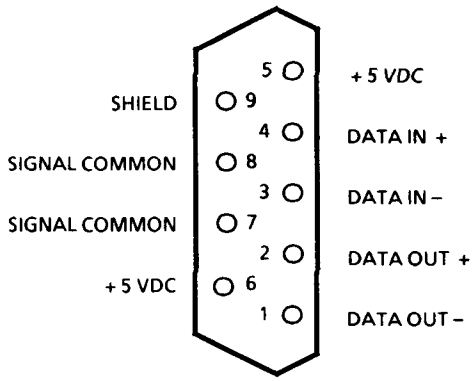


Figure 5.2 PRGMR and COMM Port Pinout

5.2 Connecting Programmers

Ports 1 and 2 support <u>all</u> SY/MAX programming devices, including the Class 8010 Hand-held Programmer. Since the Hand-held Programmer obtains its power from the Model 650, its cable is limited to six feet (2 meters) in length. Other SY/MAX programming devices have their own power supplies, and can be placed up to 10,000 feet (3050 meters) from the Model 650.

Refer to the appropriate programmer Instruction Bulletins for more details on connecting programmers to the Model 650.

5.3 Connecting Other SY/MAX Devices

Many different peripheral or programming devices can be connected to the communication ports, greatly expanding the Model 650's capability and range of applications. Figures 5.3 and 5.4 on the following pages give some possibilities for peripheral connections to communications ports 1 and 2.

Both communication ports on the Model 650 support any device that uses the SY/MAX serial protocol. This includes, but is not limited to, the following devices:

- Other SY/MAX Processors
- SY/NET Network Interface Modules
- Loader/Monitor
- Cartridge Tape Loader/Recorder
- Printers
- D-LOG Modules
- Microcell Controller
- Speech Modules

Connectable devices are described in this section. Since the *only* difference between ports 1 and 2 is in security access (Section 6), the following devices shown in Figures 5.3 and 5.4 can be connected to either communications port 1 or 2. Refer to Section 15 for information on connectable Ethernet devices.

5.3.1 OTHER PROCESSORS

Other SY/MAX Processors (Types SCP-1XX, 3XX, 4XX, 5XX, 6XX or 7XX) can be daisy-chained together, allowing them to share information and to function as an independent distributed control system. See the specific processor's Instruction Bulletin for more information.

5.3.2 SY/NET NETWORK INTERFACE

The SY/NET Network Interface Module (NIM) is a local area communications network module capable of interfacing the Model 650 to an entire network of other programmable controllers, computers, programmers, etc. Refer to NIM Instruction Bulletin 30598-257-XX for more information.

NOTE: *The Model 650 can use variable ROUTE statements for all 3 ports via Control Registers 8095, 8097 and 8098. See Section 5.5 for details.*

5.3.3 LOADER/MONITOR

The Loader/Monitor provides a simple and inexpensive method of monitoring I/O status and register values. It also allows register values to be changed without having to access the ladder diagram program.

The Loader/Monitor can display alphanumeric messages programmed in Model 650 memory, providing an inexpensive yet powerful method of signaling alarms, producing reports, and logging data. Refer to the Loader/Monitor Instruction Bulletin (30598-163-XX) for more information.

5.3.4 CARTRIDGE TAPE LOADER/RECORDER

By attaching the Cartridge Tape Loader/Recorder to the Model 650, the user's program can be stored on tape for backup purposes. Then if the user program inside the Model 650 is somehow altered or lost, the taped copy can be easily downloaded into the Model 650 as a replacement. Refer to the Cartridge Tape Loader Recorder Instruction Bulletin (30598-162-XX) for more information.

5.3.5 PRINTERS

Connecting a printer allows messages generated by the Model 650 to be recorded on hard copy. A printout ensures a permanent record of any received alarms or data. The recommended word structure for printers and other ASCII devices is to set the receiving device for 7-bits with odd parity and two stop bits. Alternate word structures are available via registers 8099 and 8100 (see Section 13.2). Refer to the Printer Instruction Bulletin 30598-176-XX or 30598-180-XX for more information.

5.3.6 D-LOG DATA CONTROLLER MODULE

The D-LOG gathers data from the Model 650 processor to generate documentation such as reports, alarm messages, and graphic displays. Refer to the Data Logger Instruction Bulletin 30598-272-XX.

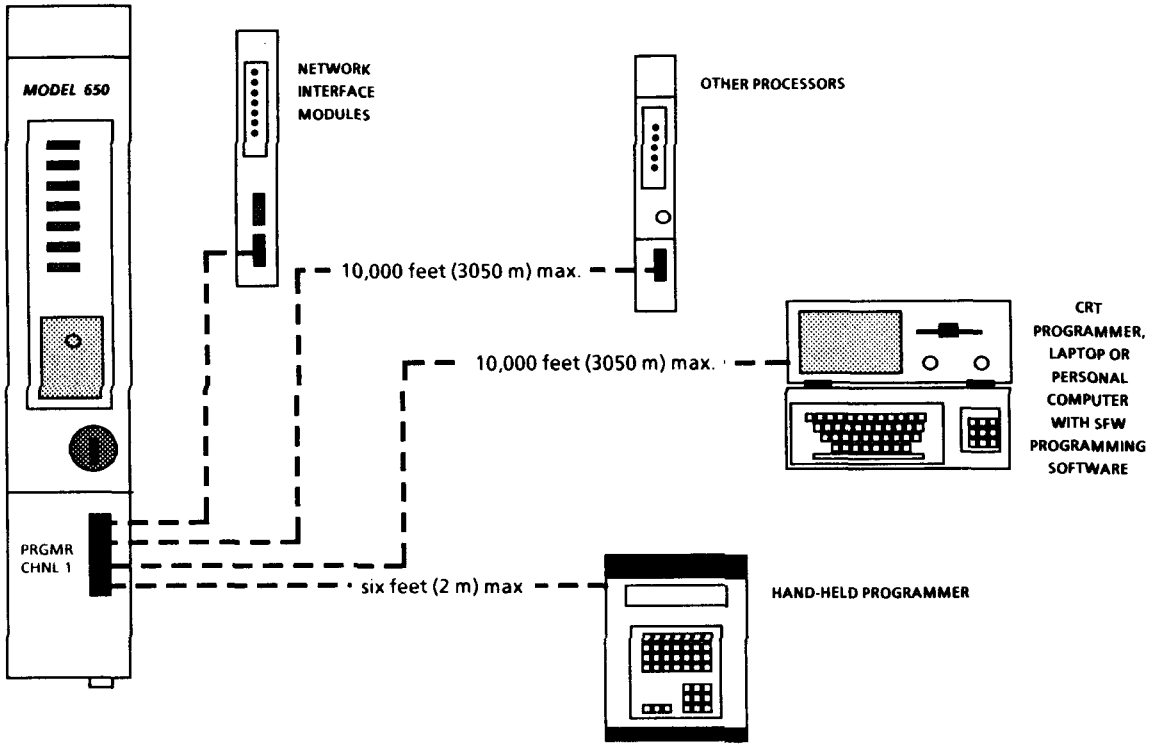


Figure 5.3 Possible PRGMR Port (Channel 1) Connections

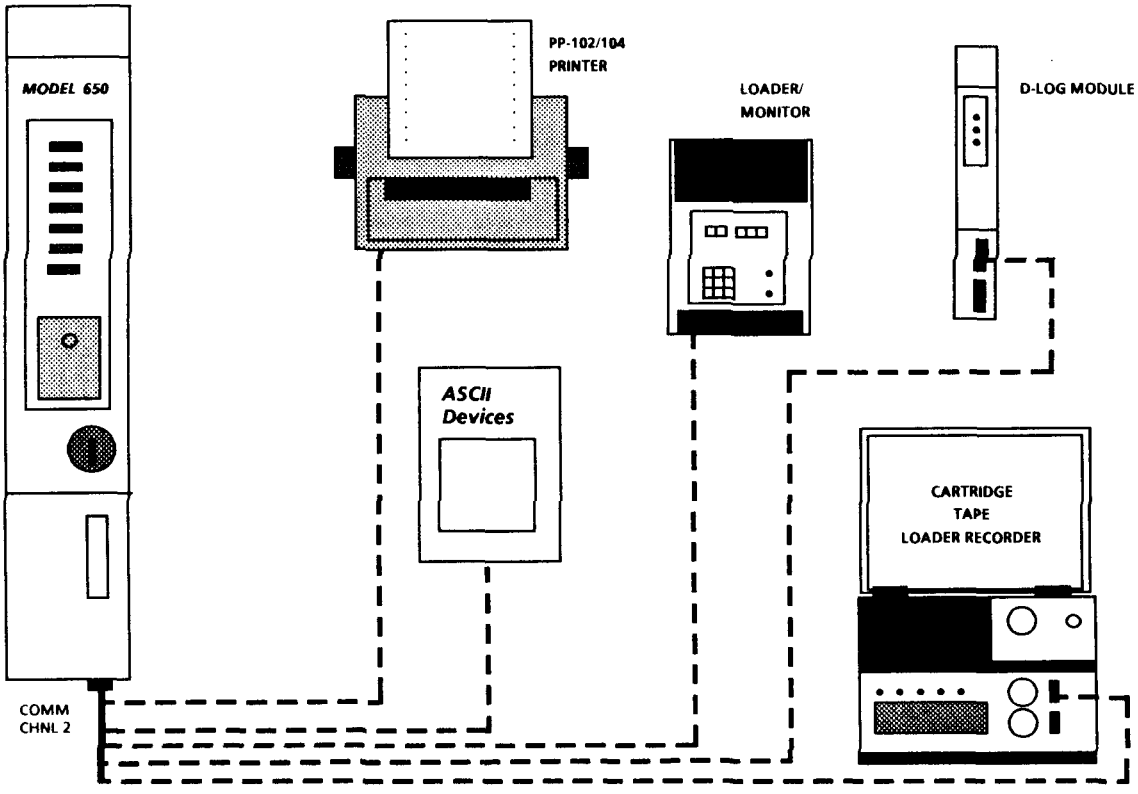


Figure 5.4 Possible COMM Port (Channel 2) Connections

5.4 Baud Rates

5.4.1 DESCRIPTION

The default communication rate for both the PRGMR and the *COMM* port is *9600* baud, which is the data rate used when the Model 650 is connected to most peripheral and programming devices. However, variable baud rates are not supported by Ethernet port 3, which operates at a fixed rate of 10 megabits /sec.

Several methods are available to change the default baud rate should it be necessary for a Model 650 port to operate at other than 9600 baud. See Figure 5.5.

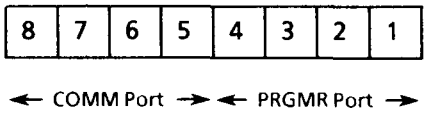
NOTE: *The BAUD rate can also be changed by using the DATA ENTER mode of a CRT, the SFW Programming Software, or a Hand-held Programmer. Enter the desired code into Model 650 control register 8169.*

OPERATION	EFFECT ON PRGMR PORT	EFFECT ON COMM PORT
PRINT command specifying baud rate	Alters baud rate	Alters baud rate
Change the contents of register 8169 using a LET command, or while in the programmer's DATA ENTER mode	Alters baud rate	Alters baud rate
DEL/CLR ALL command	No effect	No effect
Power up	Set to 9600 baud, 8-bit word with even parity and 1 stop bit.	No effect unless memory is corrupt. If so, set to 9600 baud.
Corrupt memory ("CLR MEMORY NEEDED")	Set to 9600 baud	Set to 9600 baud

Figure 5.5 Altering Baud Rates

5.4.2 ALTERING BAUD RATES

Control register 8169 in the Model 650 contains the bit patterns which determine baud rate for the communication ports. Bits 1-4 in register 8169 represent the baud rate for the PRGMR port, while bits 5-8 contain the baud rate for the COMM ports. Bit patterns are shown in Figure 5.6.



Control Register 8169 (Bits 1-8 shown)

The bit patterns corresponding to the various baud rates for both communication ports are shown in Figure 5.6 below.

The binary pattern for the PRGMR port is reflected in bits 1 to 4, while the pattern for the COMM port is in bits 5 to 8.

BAUD RATE CODE		BAUD RATE
BINARY	DECIMAL	
0000	0	75
0001	1	110
0010	2	300
0011	3	1200
0100	4	2400
0101	5	*4800*
0110	6	9600
0111	7	19.2K

Figure 5.6 Baud Rate vs Bit Pattern

Baud rates can also be changed via word attribute registers 8099 and 8100. See Section 10.

5.5 Variable ROUTE Statement

5.5.1 DESCRIPTION OF OPERATION

The Model 650 can use variables to define routing for a communication rung. All 3 ports support variable routes for communication rungs. The format of the variable communication rung is similar to a rung that has a constant route, except that a special route - **205** - is inserted just after the regular source route identifier in the rung. Three registers, 8095, 8097 and 8098, are used in conjunction with the variable route.

The generalized rung format for the variable ROUTE statement is shown below in Figure 5.7.

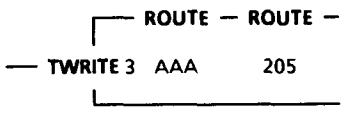


Figure 5.7 Variable ROUTE in a COMMS Rung for Port 3 (Ethernet)

When programming the communication rung that is to have the variable ROUTE, the first ROUTE in the rung (AAA) should identify the originating device. The second ROUTE must be programmed with the special number 205. The number 205 causes the Model 650 to examine either register 8095, (if the ETHERNET port is used) 8097 (if the PRGMR port is used) or 8098 (if the COMM port is used) for ROUTE data.

Registers 8095, 8097 and 8098 act as pointers to a block of from two to eight registers. The user must load the register number of the first register of this block into either register *8095*, *8097* or *8098*; each register in the block must then be preset with the intended route.

When the Model 650 executes the communication rung and encounters special ROUTE 205 as the message destination, it will check either register 8095, 8097 or 8098 to find out where the variable ROUTE information is stored. The registers being pointed to by registers 8095, 8097 and 8098 must contain the intended routes.

The block of registers containing the ROUTE information will vary in length, depending upon the need for net-to-net routing. Since the length is variable, loading the value -1 (FFFFH) into a register defines the end of the block. In the majority of cases (no net-to-net used), the block will consist of only two registers – the first contains the destination ROUTE itself while the second (containing -1) identifies the end of the block.

The user must manipulate the contents of the first register in the block to obtain the variable ROUTE. If net-to-net routing is to be used, the size of the register block simply expands to accommodate the extra ROUTE information needed. Not until -1 is encountered in a register will the Model 650 assume that all intended routing information has been provided.

Thus, since SY/MAX protocol allows for messages to accept net-to-net routing information that is up to eight levels deep, the maximum block size is seven "route registers" plus the last (terminator) register that contains -1, up to a maximum total block size of eight registers. (Note the originating route uses one of the eight levels.)

5.5.2 IMPLEMENTATION

The following steps are required to use the variable ROUTE feature in the Model 650.

1. Program a communication (COMM) rung normally, entering the number 205 as the destination ROUTE. Note that this also applies to communications that require multiple routes.

2. Prior to solving the COMM rung TRUE, load the starting register number of a block of up to eight registers into register 8095 (for the ETHERNET port) 8097 (for the PRGMR port) or 8098 (for the COMM port).

3. Also before solving the rung, preset the "route register" block with the desired routes. Signify the end of the block by loading -1 (FFFFH) into the register that immediately follows the register containing the final destination route.

4. Reroute the communication message by changing the routing specified in the contents of the register block.

5.5.3 APPLICATION CONSIDERATIONS

- The maximum count in a Model 650 communication rung is 128 registers.
- The device communicating with the Model 650 may be unable to service the full 128 registers, in which case a transmission error occurs. Following is a list of register capacities for various SY/MAX devices:

Model 100	16
Model 300	16 (15 for Series A)
Model 400	128
Model 500	128 (Series M & later), 64 (Series L & earlier).
Model 600	128
Model 650	128
Model 700	64
D-LOG Module	128
MicroCell Controller	128

- The communication statements used with variable ROUTE information are transition-sensitive.
- The status register monitors rung execution and message completion as in other COMM statements. In addition, ERROR 03 or 29 indicates an illegal rung configuration has been generated by the user (illegal route specified, illegal registers pointed to, etc.).
- The "route register block" should not extend beyond register 8000.
- The address of the status register used in the rung must be less than 8001.

5.5.4 EXAMPLE

Assume that the contents of registers 61 through 70 will be transferred *from* a networked Model 650 at location 101 *to* registers 91 through 100 in one of two processors, A and B on the SY/NET network. Processor A is at location 003. Processor B is on a separate network, at location 102, via a net-to-net link. The net-to-net link is at location 009.

The configuration of this example is shown in Figure 5.8 below. Note that Network Interface Modules (NIMs) are used with each processor.

NOTE: *All NIMs in this example must be set to the SY/MAX protocol operating mode, except for the NET-TO-NET link units.*

The steps in Section 5.5.2 are used to configure the example.

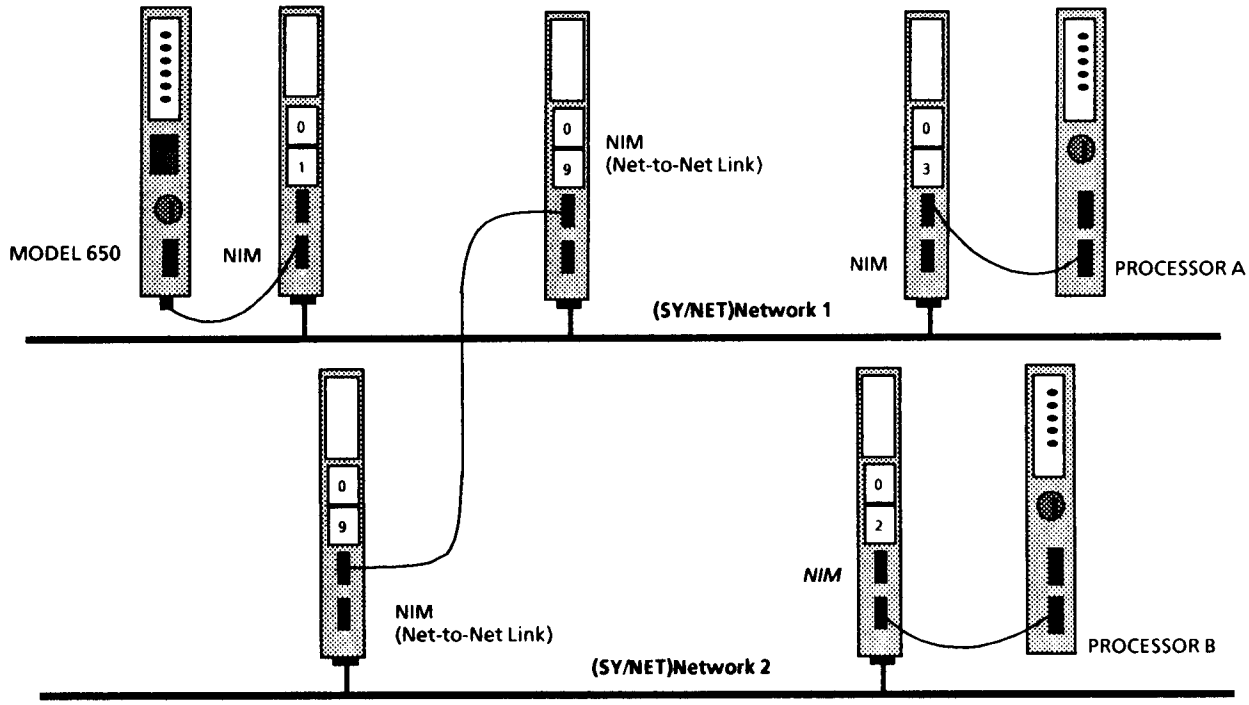


Figure 5.8 Variable ROUTE Example Configuration for Ports 1 and 2

EXAMPLE PROCEDURE

1. Program the rung shown below in Figure 5.9 in the initiating Model 650.

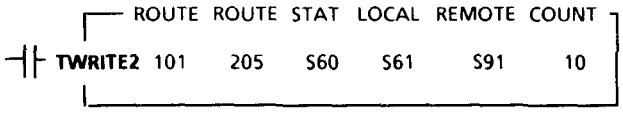


Figure 5.9 Rung for Variable ROUTE Example on Port 2

NOTE: *The assigning of register S60 as the status register is arbitrary for this example. Any unused storage register can be used.*

2. The next step is to load the variable ROUTE information into the Model 650. Do this by choosing a starting register for the "route register" block; in this example, register 3901 is used. The Model 650 uses register 8098 (COMM port) for storing this variable ROUTE information; therefore, load 3901 into register 8098. Use the DATA ENTER mode of a programming device, or a LET rung within the Model 650's ladder program, to accomplish this.

3. Assuming that we first wish to transfer registers 61 through 70 to processor A, the value 0003 (processor A's location) must be loaded into register 3901. To terminate the block, load the value -0001 into register 3902 to show the Model 650 the end of the "route register" block. Again, a programming device in the DATA ENTER mode or a LET rung is used for this.

   When enabled by the logic, the communication statement is transmitted out of the Model 650's channel 2 (COMM) port through the NIM which is set as location 101.

When the logic in this communication rung is enabled, the Model 650 executes the rung with the *following results:*

- The ROUTE 205 in the rung causes the Model 650 to interrogate register 8098 for ROUTE information.
- Register 8098 (containing the value 3901) directs the Model 650 to examine register 3901.
- Since register 3901 = 3, the value 003 becomes the ROUTE destination.
- *The Model 650 next interrogates register* 3902. Since 3902 = -1, the Model 650 knows it has all the necessary ROUTE information. The contents of registers 61 through 70 are now sent to processor A's registers 91 through 100.

4. Now assume the register block 61 to 70 is to go to processor B's registers 91 through 100 instead. Since transmission to processor B involves a net-to-net link, an additional ROUTE parameter is needed. Enter the following values into the indicated registers:

   Register **3901** . . . . . **0009** (net-to-net link)

   Register **3902** . . . . . **0102** (processor B's location)

   Register **3903** . . . . . **-0001** (end ROUTE information)

   When the logic in the communication rung is enabled, registers 61 through 70 in the Model 650 are now routed through location 0009 to processor B at location 102.

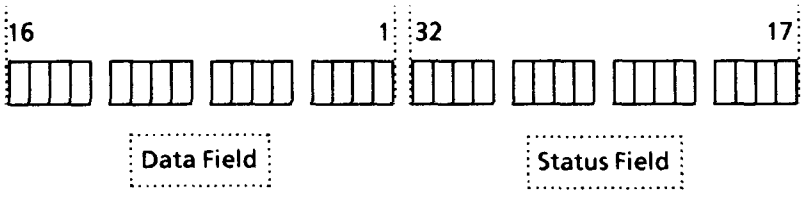
5.6 Miscellaneous Considerations

Because the primary function of a programmable logic controller is the timely execution of the user program, communication with external devices is subject to certain rules and limitations. The following is a discussion of some of the potential pitfalls associated with these rules, as well as some techniques for improving communications throughput.

Both the sending and receiving of communication characters in READ, WRITE, ALARM, and PRINT (ASCII input/output) statements can occur at any time while the processor is scanning. Typically, the impact on scan time for servicing these characters is slight because of separate microprocessors used for control and communications. The largest single delay associated with the communication process occurs when a complete message has been received, and the register data contained in that message must be incorporated into the processor's image table registers. From 2.5 to 5.0 msec is allowed for the processor at the end of scan (EOS) to perform this task before scanning must resume. Note, processors which are in HALT can still respond to incoming messages, but cannot initiate them.

5.6.1 IMPROVING COMMUNICATION THROUGHPUT

The maximum block size of registers (count) which a user may specify in a message is 128 registers. For communicating a large quantity of registers, throughput is optimized by transmitting the largest blocks possible; a single transmission with 128 registers is over 10 times faster than 128 transmissions of 1 register each. The larger the message block, however, the more time is required at EOS to incorporate this data into the processor's image table. Because this processing time for 128-register messages can exceed 5 msec, multiple scans may be required to update all the registers. This presents a potential application problem if the new data needs to be treated as a contiguous block for control purposes. A simple example is floating point data, in which an odd/even pair of registers constitutes one floating point value, and splitting of this floating point register pair could result in an invalid value used by the PLC.



Bits	Description
1-14	Error Codes (Always Odd - bit 1 set)
15	Remote Device in RUN
16	Successful Completion
17	Message Sent to Buffer
18-21	SY/MAX Timer Value
22	Input Ladder Logic Solved True
23	SY/MAX Timeout Occurred
24-32	*Register Identification Code (N/A)*

Note: *Bits 1-23 are cleared when bit 22 is reset (input logic to Communications rung opens up)*

Figure 5.10 Communication Rung (READ, WRITE, PRINT, ALARM) SY/MAX Status Register

To prevent this potential "splitting" of a message over multiple scans, bit 16 of register 8170 may be set by the user. Setting this bit causes the processor to *completely* finish servicing a message at EOS prior to resuming scanning the user program; thus, the 2.5 to 5.0 msec EOS period is extended for as long as necessary, with worst-case extension being 10 msec total. Therefore, 8170-16 should be set any time multiple registers in a message must be updated during the same scan (as with floating point data) with the trade-off being a potential 5 additional msec tacked on to the present 5 msec EOS time.

In the event a contiguous block of data exceeds 128 registers, application programming techniques must be used to inhibit control logic operation until all the new data has been incorporated into registers. In the initiating processor, bit 16 of the associated communication rung status register is set when an operation is complete; thus, multiple bit 16's may be monitored to ensure that each associated message has been completed. Refer to Figure 5.10 for an overview of bit descriptions. However, when unsolicited WRITE commands are initiated by a remote device, the local device is not advised when an operation is complete. Therefore, the remote device should use some of the transmitted registers to advise the local device. One technique is to increment the first and last register of the contiguous block each time a block of messages are sent; the local device would then need to be programmed to only act upon the data when these registers were equal, indicating that we are not in the middle of an update and the data can be considered valid.

Another way of improving communications throughput by trading communication servicing time for scan time is the use of immediate communications update coils in the user program. Each time an immediate communications update coil (bits 8176-11 through 8176-13 for ports 1 through 3, respectively) is encountered, the processor checks the indicated port for a received message. If one exists, it will be serviced for up to 5ms, thus imposing a worst-case scan time impact of 5 msec. For applications in which it is desirable to trade scan time for improved communications throughput on port 3, for example, coil 8176-13 may be used as often as desired at multiple program locations to process existing messages.

5.6.2 SY/MAX COMMUNICATION TIMEOUT

As discussed in the programming bulletin, bits of the communication status register (which must be uniquely assigned to the communication statement) can be used to monitor the communication process. For example, bit 22 is set to 1 when the input logic to the rung is true. Bit 17 is set to 1 when the message has been loaded into a commmunication buffer and is about to be transmitted. And bit 16 is set to 1 when a reply is received indicating the message transaction is successfully completed. (Bit 1, along with some combination of bits 2 through 14, is set to indicate an error and indicate an unsuccessful message transaction). Finally, bit 15, when set, indicates the remote device is actively scanning logic or executing a program.

Another feature available to the user is a communication reply timeout flag. This differs from the Ethernet timeout discussed in Section 15.6.2 in several ways. First, it can be applied to communications through all 3 ports. More importantly, this timeout is based on message *completion*, not just message acknowledgment. For a READ statement, for example, timing begins when the message is loaded into a communication buffer (bit 17 set to 1) and ends when the requested data has actually been received (bit 16 set to 1) or when time has expired or an error reply was received (bit 1 set to 1). Bit 23 of the communication status register is set any time a complete message response is not received prior to a user-specified period after the message has been initiated. This period is determined by the bit pattern contained in the appropriate bits of register 8168 at the time the communication rung is first solved true. If the time period selected in register 8168 is too short, the reply could still arrive after the timeout (bit 23) is set and the appropriate registers updated.

8168 Bits 1-4, 5-8 and 9-12:

BIT PATTERN (bits 4-1 for PRGMR bits 8-5 for COMM bits 12-9 for ETHERNET)	TIME COUNT IN MILLISECONDS
0000	0 to 250
0001	250 to 500
0010	500 to 750
0011	750 to 1000
0100	1000 to 1250
0101	1250 to 1500
0110	1500 to 1750
0111	1750 to 2000
1000	0 to 2000
1001	2000 to 4000
1010	4000 to 6000
1011	6000 to 8000
1100	8000 to 10000
1101	10000 to 12000
1110	12000 to 14000
1111	14000 to 16000

Figure 5.11 Communication Timeout Ranges

As can be seen from Figure 5.11, two ranges are available: 0 to 2 seconds (with 250 msec resolution) or 0 to 16 seconds (with 2 sec resolution). Note the time period associated with a bit pattern is an interval, with the timeout liable to occur anywhere within the interval. For example, a bit pattern of "0000" can result in a timeout being flagged anywhere from 0 to 250 milliseconds (maximum) after a message has been sent to the buffer.

When a message is initiated, the user-specified timeout code for that channel is copied into bits 18 to 21 of the corresponding communication status register. These bits then "count down " the time until a reply is received. If a reply is not received (bit 16 or bit 1 set to 1) within the timeout period, bit 23 will be set.

Note: *Each time the comms rung is solved false, all bits associated with the commmunication status register are cleared.*

COMM STATUS BIT DEFINITIONS:

Bit 16 only is set: A valid reply has been received; message *complete before timeout has* elapsed.

Bits 16 and 23 are set: A valid reply was received; however, *after* the timeout had elapsed.

Bit 23 only is set: No reply received at all, timeout has elapsed.

Bits 1 and 23 are set: A valid *error* reply was received either before or after timeout. Check the data field for the error code.

CONSIDERATIONS FOR USING TIMEOUT

- Bits 1 through 23 in the status register are cleared every time the rung is executed FALSE.
- If a message is not acknowledged (no reply), ERROR 13 is posted in the status register and a processor communication port error is sent to register 8175 and bit 23 is set. For ports 1 and 2, this may occur when cabling to the ports is disconnected. For port 3, this may occur if the Ethernet NIM portion has shutdown---a fatal Ethernet communications error.
- If an error reply is received, the appropriate error code is posted in the status register and bit 23 is set, regardless of the actual time elapsed.
- If a valid reply is received after a timeout has occurred and bit 23 is set, bit 16 is set and bit 23 is latched ON until the communication rung is executed FALSE.

EXAMPLE: A user wishes to know if a certain ladder-initiated Ethernet message is receiving a reply within four seconds after initiation. The bit pattern **1001** should be loaded into register 8168 bits 9 through 12 prior to the communication rung being solved true (for checking Ethernet port timeout).

If a reply is not received within 4 seconds (4000 milliseconds - the upper timeout limit), bit 23 of the appropriate status register is set. Keep in mind the bit can be set anytime after 2 seconds has elapsed since this is the lower timeout limit.

5.7 Technical Data For Communication Ports

TECHNICAL CHARACTERISTIC	APPLICABLE PORT		
	1	2	3
Connector Type			
D-type, 9-pin female slide lock (RS-422)	X	X	
ThinWire Ethernet- BNC, with gold-plated conductor			X
Communication Method			
RS-422 (serial differential)	X	X	
High speed Ethernet			X
Communication Rates			
(See Figure 5.5 for BAUD rate information)	X	X	
10 megabits/sec			X
Maximum Cable Length			
10,000 ft. (3050m) at 9600 baud (RS-422)*	X	X	
Ethernet-maximum:** 607 ft. (185 m) per segment, 3034 ft. (925m or five 185m segments) with repeaters			X
Cable Type			
Belden® #8723 or equivalent (RS-422)	X	X	
Ethernet-type RG58A/U or RG58C/U 50-ohm coaxial cable			X
Short Circuit Protection			
Driver/receiver circuitry	X	X	
Ethernet			No
Surge Protection			
Driver/receiver circuitry	X	X	
Ethernet			X
Common Mode Voltage Rating			
6 volts maximum	X	X	N/A
Auxiliary Power			
Powers the hand-held programmer or loader/monitor	X	X	N/A

Figure 5.12 Communication Ports Technical Data.

* *Actual cable length for RS-422 communication is subject to installation considerations such as reliable building ground (signal reference), common mode voltage, signal interference/shielding, etc. If communication problems are experienced over long cable runs, decreasing the baud rate may result in greater signal integrity.*

** *Applies to 10BASE2 ThinWire Ethernet networks. Longer distances are possible using repeaters with other media such as 10BASE5 Standard Ethernet.*

6 SECURITY FEATURES

6.1 Description

The Model 650 processor has security features that prevent unauthorized access to user ladder programs, labels, rack addressing and storage registers. This type of security protects programming equipment and/or other devices connected directly to the processor via SY/NET or via Ethernet (Model 650 or SFW390 software).

NOTE: *Except where otherwise noted, this section refers to various security features for all 3 ports (PRGMR, COMM and Ethernet ports---Channels 1, 2 and 3).*

Three types of security features are used to protect each of the following:

1. Ladder memory
2. Data storage registers
3. Communications

Section 6.3 contains three tables that summarize all of the available security features in the Model 650 processor.

6.2 Definitions

The following terms are used frequently in this section.

Editing ..... The use of external equipment to insert, replace, delete or clear rungs, rack addressing, and labels that exist in the processor memory.

Register ..... Storage of I/O status, data values, or control information. Can be either *data* registers or *control* registers.

Non-Priority Register Read .. Access to registers to display or copy (read-only) a value stored in a Model 650 data register. Non-priority register reads are normally done through programming equipment and software packages (i.e. Class 8055 and Type SFW-390).

Non-Priority Register Write . To enter or store a new value in a Model 650 data register. Non-priority reads are normally done through programming equipment and software packages (i.e. Class 8055 and Type SFW-390).

Priority Register Read ..... To read or recall the contents of one or more SY/MAX registers. Typically this type of communication occurs between processors and software packages (i.e. Class 8055 and Type SFW-390).

Priority Register Write ..... To write or store a value into a SY/MAX register. Typically this type of communication occurs between processors and software packages (i.e. Class 8055 and Type SFW-390).

Programming Device ..... SY/MAX CRT (SPR-200, 250, 300) SY/MAX hand held programmer (SPR-100), loader monitor (SLM-100), or an IBM-Compatible computer equipped with SY/MAX Type SFW Programming Package.

User Memory .. The memory in which user-created ladder programs, rack addressing, and labels are stored.

Storage Register Memory ..... Memory where data register values are stored in the Model 650, separate from user memory.

6.3 Summary

The three tables on this page describe the ladder, register and communication security features available in the Model 650 processor:

- Figure 6.1 covers ladder program security features.
- Figure 6.2 covers storage register security features.
- Figure 6.3 covers communication security features.

In the following tables, the *method of protection* is indicated in the left column. *What the method protects* is shown in the center column. The right column gives the *subsection of Section 6* where that protection feature is described.

SECURITY METHOD	PROHIBITS	SEE SECTION
Hardware Jumper	Program editing of *any* kind.	6.4
Port/Route Edit Lockout	Program editing by any port or route other than the one used by the initiating equipment.	6.5
Keyswitch RUN position	Program editing of any kind, including forcing.	6.6
Inhibit Rung	Altering rungs within an area defined by the inhibit rungs.	6.7
Safeguard Rung	Displaying, by programming equipment, the ladder areas defined by the Inhibit Rungs.	6.8
Password/ Restriction Register	Editing on a port-by-port basis.	6.9
Memory Protect Bit	Ladder programming or editing*	6.10

* Does not apply to PRGMR port

Figure 6.1 Ladder Program Security Methods

SECURITY METHOD	PROHIBITS	SEE SECTION
Password and Restriction Registers 8177 and 8178	• External non-programming devices from altering registers on a port-by-port basis. • Programming equipment from altering register data on a port-by-port basis • Forcing of I/O by programming equipment on a port-by-port basis	6.9
Protect All Registers Bit	Altering of register data by any external device*	6.12
Force Inhibit Bit	Forcing of I/O by programming equipment*	6.11
Safeguard Rung	Programming rungs containing protected registers	6.8
Fence Registers	Altering of registers that are defined by user as fenced*	6.13

* Does not apply to PRGMR port

Figure 6.2 Register Data Security Methods

SECURITY METHOD	PROHIBITS	SEE SECTION
Safeguard Rung	The programming of COMM rungs (READ, WRITE, PRINT, ALARM) assigned to user-specified channels on a port-by-port basis	6.8

Figure 6.3 Communication Security Methods

6.4 Hardware Security Jumper

Prohibits: Ladder editing or programming of any kind through any of the 3 ports (COMM, PRGMR or Ethernet).

Uses: Internal jumper, located on a three-pin header. The jumper is accessed by removing the Model 650's side plate. Refer to Figure 6.4.

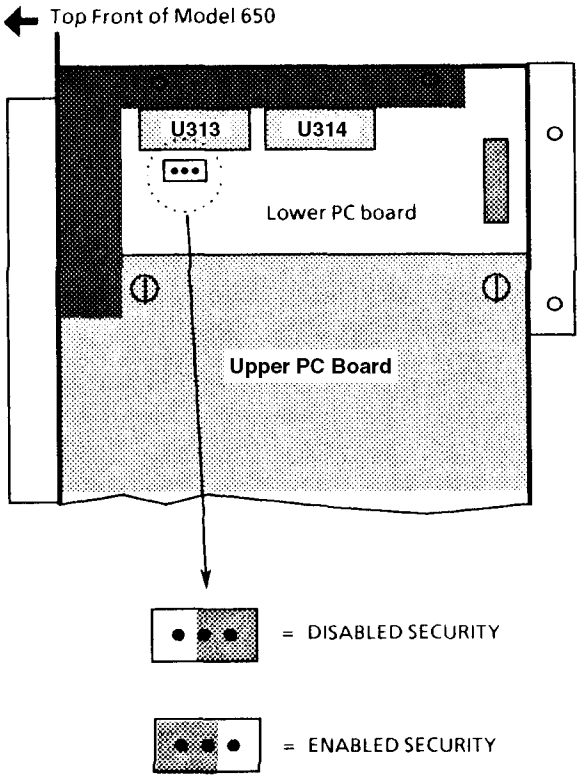


Figure 6.4 Hardware Security Jumper

The security jumper bridges the center pin of the three-pin header to *either* the left or right pin (see Figure 6.4). When the security jumper is in the ENABLED position, all ladder memory is protected from alteration of any kind. When DISABLED, ladder memory is accessible to the extent that other security methods allow. The WRITE PROTECT LED (Section 4.2) illuminates when the jumper is enabled.

CONSIDERATIONS

- The CLEAR ALL command will have no effect when the jumper is enabled.
- An enabled jumper overrides the following security features:
  1. Inhibit Rung (Described in Section 6.7)
  2. Memory Protect Bit (Section 6.10)
  3. Password & Restriction Register features for selective disabling of ports (Section 6.9)

SETTING THE SECURITY JUMPER

CAUTION

Remove power from the rack before removing the Model 650 processor.

1. Turn off the AC power to the rack where the Model 650 is installed.

CAUTION

Static damage may occur if board is not handled properly.

2. Remove the Model 650 from its rack.
3. Remove the side access plate from the Model 650.
4. Using a needlenose pliers, place jumper in the desired position (ENABLED or DISABLED) as shown in Figure 6.4. DO NOT LEAVE THE JUMPER OFF OF THE HEADER. It must be in one of the two positions.
5. Replace the side cover on the Model 650, and reinsert it in its rack.
6. Reapply rack power.

6.5 Selective Port & Route Edit Lockout

Prohibits: Ladder program alteration through any port or by any route *except the port and route used by the initiating programming device.*

Uses: Control Register 8176, bit 15.

This security feature ensures that only a single programming device on the SY/NET network has access to the Model 650's memory at any given time. If the SY/NET network is not in use, the restriction applies channel-by-channel.

To engage the Port/Route Lockout feature, set bit 15 of Control Register 8176 to "1" using the Data Enter mode of the programming device. The Model 650 "remembers" the port and route that the programming device used to set bit 15. Any attempt to edit the program through any other port or route other than the "remembered" port/route causes the Model 650 to generate an ERROR 49.

The lockout feature does *not* prohibit simple interrogation or viewing of the ladder program, unless other security features that prevent these are active. *Register* contents can still be altered (unless other measures are active to prevent this). To release the lockout, use the programmer that set bit 15 of Register 8176 to "1" (the same route/port) to reset the bit to "0".

CONSIDERATIONS

- A CLEAR ALL command issued through the PRGMR port of a Model 650 that has Port/Route Lockout enabled has no effect *unless* the device sending the CLEAR ALL is the same device that enabled the lockout.
- Cycling power to the Model 650 will disable the lockout and extinguish the WRITE PROTECT LED (if no other form of security is active).
- If the lockout is disabled by cycling power, bit 15 of Control Register 8176 may remain set at 1. This is because the processor retains Control Register contents, but *cannot* retain port and route information, when power is removed. Thus, the fact that bit 15 of 8176 is set at 1 is not a guarantee that the lockout feature is active. Conversely, if bit 15 is set to 0, the lockout feature may still be active. Only by setting the bit *immediately prior* to editing, and resetting the bit *immediately after* editing can bit indication be considered reliable.

6.6 Keyswitch RUN Position

Placing the Model 650 keyswitch in the RUN position prohibits any changes to be made in a ladder program. In addition, the ability to force, or to alter forcing, is disabled while the keyswitch is in RUN (if changes to a program that is running are to be made, place the keyswitch in the RUN/PROGRAM position).

6.7 Inhibit Rung

Prohibits: Altering rungs within protected ladder areas that are defined by the Inhibit Rungs.

Uses: Control Register 8176, bit 16

A maximum of two areas of the user ladder program can be protected from being edited. The two areas that make up a typical ladder program are the *MAINLINE* ladder area and the *SUBROUTINE* ladder area.

The *subroutine* area consists of all the subroutines that exist in the user ladder program. This area starts with the MARK ST SUB rung and proceeds to the last rung in memory. The *mainline* ladder area consists of rung number 1 in memory and proceeds up to, but not including, the MARK START SUB rung.

Protected *mainline* ladder rungs always start with rung number 1 in the ladder and consecutively proceed up to the inhibit rung in the mainline area.

The protected *subroutine* rungs always start with the MK ST SUB rung and proceed consecutively to the inhibit rung in that area.

An inhibit rung is inserted into a program by entering a rung containing only a coil (no conditional logic) addressed as *register 8176 bit 16.* See Figure 6.5, below:

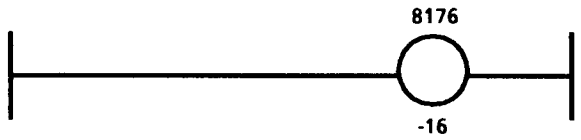


Figure 6.5 Inhibit Rung

CONSIDERATIONS

1. Protection is not enabled until either the processor is power-cycled or the keyswitch is turned from HALT to RUN.

2. More than one inhibit rung may exist in any given area, but only the earliest (lowest number) programmed inhibit rung in an area defines the end of protected ladder.

3. When a CLEAR ALL command is issued to the processor over channel 1, the protected areas defined by the inhibit rungs will *not* be cleared. Only the unprotected areas of the ladder program will be cleared.

CLEARING THE INHIBIT RUNG

Clearing the protected and unprotected areas of a ladder program that are protected by Inhibit Rungs requires that the *entire* ladder program and all rack addressing be cleared from the Model 650. This is done by connecting a programmer to the PRGMR (channel 1) port, and then performing the following steps:

1. Confirm that the security jumper (Section 6.4) is in the DISABLED position.

2. Access the RACK ADDRESSING mode.

3. Press the DELETE soft key.

4. Press the CLEAR ALL soft key *twice*.

5. Access the DATA ENTER mode.

6. Store the decimal value **-1** in register 8177 (the password register).

7. Repeat steps 2 through 6.

The result of the above CLEAR ALL operation is a completely blank user memory, meaning that:

- No Ladder Program
- Default Rack Addressing
- Zeroed Register Values
- No Labels
- No Enabled Security Features

...will exist within the Model 650.

Figure 6.6 illustrates the operation of the Inhibit Rung.

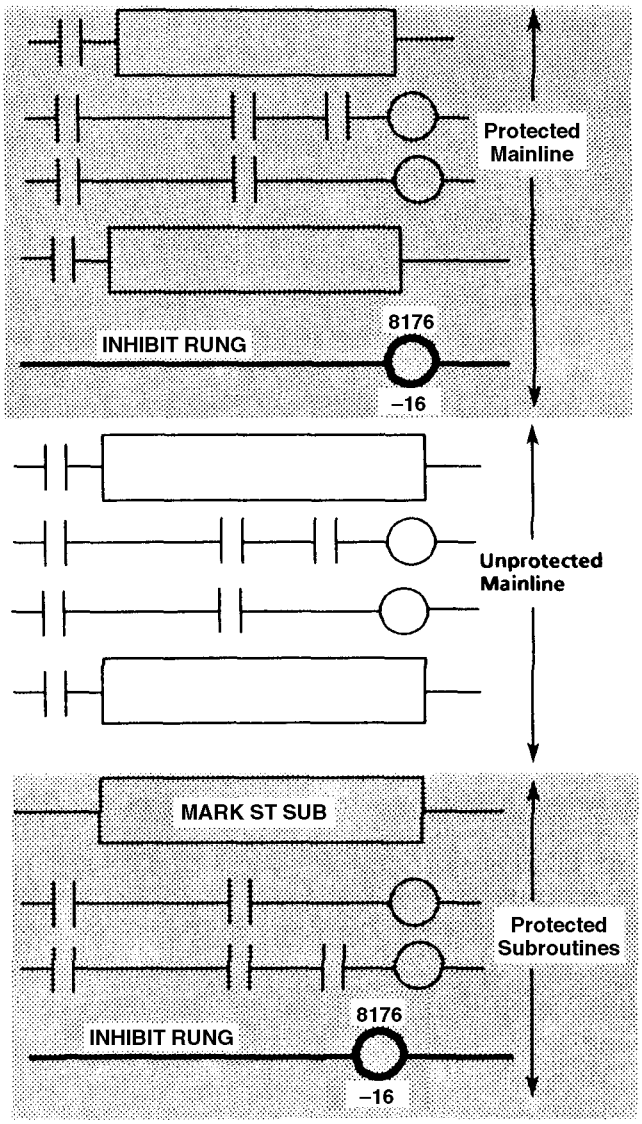


Figure 6.6 Operation of the Inhibit Rung Coil

6.8 Safeguard Rung

Prohibits:

1. Displaying, by programming equipment, any rungs protected by Inhibit Rungs (see Figure 6.6). Two different groups, each with an Inhibit Rung, can be protected. Enabled protection applies to both channels.
2. The programming of any COMM rungs (READ, WRITE, PRINT, ALARM) assigned to user-specified channels on a channel-by-channel basis.
3. Programming any rungs that contain registers which are user-defined as protected.

Uses:

1. Inhibit Rung with Safeguard Rung, Parameter A below (#1 and 2 above).
2. Inhibit Rung with Safeguard Rung, Parameters B and C below (#3 above).

The safeguard rung allows a variety of security protection features to be enabled. To use the safeguard features the special TLET rung as described below must be programmed as the first rung in ladder memory. To activate the **A, B_1, B_2, C_1, and C_2,** parameters in the special TLET rung, an inhibit rung (with coil addressed as 8176-16) must also be programmed.

The following rung **must be** rung #1:

—[TLET S8165 = A ; B_1 ; B_2 ; C_1 ; C_2 ; D ; E_1 ; E_2]—

Where:

A is port safeguard code for COMM-rung/view inhibit.

B_1 is *STARTING* ADDRESS for Group 1 Safeguarded Registers.

B_2 is *ENDING* ADDRESS for Group 1 Safeguarded Registers.

C_1 is *STARTING* ADDRESS for Group 2 Safeguarded Registers.

C_2 is *ENDING* ADDRESS for Group 2 Safeguarded Registers.

D is the *SCAN TIME LIMIT* in milliseconds

E_1 is *TOLERANCE OF SCAN INTERRUPT* in 2.5 millisecond counts.

E_2 is *RATE OF SCAN INTERRUPT* in 2.5 millisecond counts

The parameters above are explained in detail below:

A A number entered here (000 through 007) prevents Channel 1, Channel 2 and/or Channel 3 ports from being assigned to communication rungs (READ, WRITE, PRINT, ALARM). Figure 6.7 shows which number restricts what port(s):

CODE	ETHERNET Port Restricted	COMM Port Restricted	PRGMR Port Restricted
X00	NO	NO	NO
X01	NO	NO	YES
X02	NO	YES	NO
X03	NO	YES	YES
X04	YES	NO	NO
X05	YES	NO	YES
X06	YES	YES	NO
X07	YES	YES	YES

Figure 6.7 Port Safeguard Codes

When "1" is substituted for "X" as the first digit in the number (i.e., 100, 101, etc. through 107), the ability to view, display, record or print the rungs through all ports is also prohibited.

EXAMPLE: To prevent only the PRGMR port (channel 1) from being assigned to communication rungs ANDto inhibit the ability to view, display, record, or print out the program through *any* port, enter the number "101" as the port safeguard code.

Safeguarding *registers* prohibits those registers from being programmed into counters, timers, or shift registers. Also, safeguarded registers cannot be used to the left of an equal sign in a LET statement, or have their bits programmed as coils.

Safeguarded registers are also protected from being written to by other SY/MAX processors *unless* the other processor's WRITE communication rungs are also protected from within an Inhibit Rung ladder area.

B_1 *Starting* address for the *1st group* of safeguarded registers. The starting address can be any value from 1 to 8192.

B_2 *Ending* address for the 1st group of safeguarded registers. This address must be greater than or equal to **B_1**.

B_1, **B_2**, **C_1**, and **C_2** *must* be separated by a softkey-generated semicolon.

C_1 *Starting* address for the *2nd group* of safeguarded registers. The starting address can be any value from 1 to 8192.

C_2 *Ending* address for the 2nd group of safeguarded registers. Must be greater than or equal to **C_1**

D *Scan time limit* in milliseconds. If the Model 650 exceeds the scan time limit, it will halt with "ERROR 970" in error register 8175. When **D** is "0", a default scan time limit of 1 second is assigned.

SAFEGUARD RUNG CONSIDERATIONS

- Zeros should be placed in parameter locations for features that are unused. For example, if only one group of registers is to be protected, enter "0" for the C_1 and C_2 parameters.
- If Timed Interrupt tolerance and rate are the only parameters used (no Safeguard parameters), only values for E1 and E2 need be entered. Because the Safeguard rung parameters are read from right-to-left, E1 and E2 will be the first parameters encountered and are interpreted as Timed Interrupt parameters. Omitting values for other parameters causes them to be ignored.
- *Only* register 8165 can be assigned as the safeguard TLET rung.
- Only one safeguard TLET rung is allowed in the entire ladder program, and it must be the *first* rung in the program.
- Make sure the Model 650 is in HALT before entering the Safeguard rung into the program. If the Model 650 is in RUN while the rung is entered, the rung will execute normally and possibly create unwanted values in registers 8165, 8166, 8167, and any others used in the rung.
- Parameters A;B1;B2;C1;C2 of the TLET rung require that an inhibit rung exists in the ladder program. If no inhibit rung exists, the safeguard parameters A;B1;B2;C1;C2 are ignored.
- The safeguard TLET rung parameters cannot be altered once the inhibit rung is programmed and the Model 650 is power-cycled or is keyswitched to RUN. *All ladder program and rack addressing must be deleted before the safeguard rung can be altered.*
- If password register 8177 is included in the block of safeguarded registers, the only way to override and delete the safeguard rung is to remove the Model 650 from the rack, and then remove the onboard battery. This allows the memory to completely discharge.
- To override and delete the safeguard rung, the entire ladder program and all rack addressing must be cleared from Model 650 memory. This is accomplished by connecting programming equipment to channel 1 (PRGMR Port), and performing the following steps:

  1. Confirm that the Security Jumper is in the DISABLED position (Section 6.4).
  2. Access the RACK ADDRESSING mode.
  3. Press the DELETE soft key.
  4. Press the CLEAR ALL soft key *twice*.
  5. Access the DATA ENTER mode.
  6. Store the decimal value **–1** into register 8177 (the password register).
  7. Repeat steps 2 through 6.

The result of the CLEAR ALL operation is a completely blank user memory, meaning that...

- No Ladder Program
- Default Rack Addressing
- Zeroed Register Values
- No Labels
- No Enabled Security Features

.... will exist within the Model 650.

NOTE: *The Rack Address CLEAR ALL operation clears or resets all processor registers. However, the Ethernet NIM registers are not cleared. Those registers not cleared or reset include the ENIM storage (mailbox) registers (regs. 3000-6999) and the Ethernet communication performance data registers (regs. 7000-7999). Two alternatives for clearing or resetting the ENIM registers exist:*

1. *Remove the Model 650 from the SY/MAX rack, remove the battery form the Model 650 and wait for approximately 2 minutes for the memory to clear. Then replace the battery, place the Model 650 in the rack and provide power.*

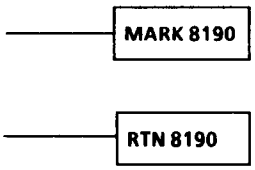
2. *The second method is to program a series of WRITE rungs into the Programmable Controller to clear the ENIM storage registers (3000-6999). Cycling power then clears the performance calculations (7XX2-7XX5), but not the performance parameters (7XX0-7XX1) or the storage registers. Developing a small utility ladder program to clear the storage registers and set the default Ethernet parameters may be useful. This program could then be run anytime a Rack Address CLEAR ALL is issued to clear or reset all the desired registers.*

6.8.1 TIMED INTERRUPT OPERATION

The parameters E_1 and E_2 in the TLET S8165 rung define the values for timed interrupt *tolerance* and timed interrupt *rate*, respectively.

This method of scan control interrupts the ladder program at specified time intervals to execute a certain group of rungs (a special subroutine), and then returns to the main program. This method of interrupt is useful for applications where certain inputs of data must be *continuously sampled within a particular repeatable time limit.*

Subroutine number 8190 is reserved for the timed interrupt function. The following two rungs are used to identify the timed interrupt subroutine:



Refer to the "Processor Scan Control" section in the respective programmer manual for an in-depth discussion of using GOTO, GOSUB, and TIMED INTERRUPT scan controlling techniques.

The *purpose* of the timed interrupt is to ensure that, regardless of program length, a specific block of rungs is executed within a given period of time. Typically this involves monitoring a select group of *inputs* while programmed rungs control a select group of *outputs*.

The timed interrupt subroutine should *not* contain any transition-sensitive rungs, or any rungs that require multiple program scans to complete their operation. Examples of these rungs include communication statements (READ, ALARM, WRITE), transitional coils or LETs (TLETs), shift registers, and timers.

While the programmed interrupt rate specifies the rate at which the interrupt is to occur, it may not always be possible to interrupt the Model 650's scan *precisely* at that rate. For example, the processor cannot be interrupted while it is scanning a rung or during end-of-scan activities such as updating external I/O and servicing communication ports.

NOTE: *RUN mode programming is not allowed when the TIMED INTERRUPT is enabled; attempts will be rejected with ERROR 79 posted by the programmer.*

In order to be assured that the processor has sufficient time to call the timed interrupt subroutine, the programmed tolerance must exceed the external I/O update time plus the communication port servicing time. The time required to update external I/O is a function of system layout and rack addressing; refer to Section 14.2 for more information regarding I/O updating. Servicing the

communication ports can require an additional 5 milliseconds per scan if saturated. These factors can combine to form a substantial time period during which the Model 650's scan *cannot* be interrupted.

> **CAUTION**
>
> **A program containing a Timed Interrupt should ALWAYS be tested first, under conditions that simulate the rack addressing and communication demands that the system will experience when *installed at the actual job site.***

In the event that interrupt tolerance errors (ERROR 29102) are encountered, it may be possible to eliminate these errors by optimizing rack addressing and/or restricting communication traffic. If errors persist, the programmed tolerance value needs to be increased.

In the majority of applications that require speed, the inherently fast scan speed of the Model 650 eliminates the need for the timed interrupt function.

NOTE: *Since the Model 650 updates external I/O at the end of each ladder scan, any timed interrupt that is executed more than once per scan must also update the external I/O at the same rate as the timed interrupt occurs, as discussed in the following.*

Registers 8105 and 8106, in conjunction with bit 7 of register 8176, can be programmed to perform an immediate I/O update within the timed interrupt (see Section 13.2). This selectively updates any registers that require fresh information. Thus, registers used as *inputs* within the timed interrupt subroutine should be updated at the *beginning* of the subroutine, while *output registers within the* subroutine should be updated *just prior to exiting* the subroutine.

A scan penalty is paid to accomplish immediate I/O updating, and must be considered when determining the execution time of the timed interrupt subroutine. Refer to Figure 14.3. The time needed for this updating becomes more and more significant when the ladder portion of the subroutine is short. To economize these register updates, group together input and output addresses that are used in the timed interrupt to minimize the scan time impact of performing immediate I/O updates.

6.9 Password and Restriction Registers

The Model 650 processor uses register 8178 for storing an access code and register 8177 for storing a "password" value.

The following restrictions can be imposed on either port by using Password Register Restriction:

1. **Inhibit ladder programming or editing.** Restricts the programming or editing of ladder logic using any programming device. *Bit 4, 8 or 12 of register 8178.*

2. **Prevent programming equipment from forcing I/O registers.** Restricts the forcing of register outputs using programming equipment. *Bit 2, 6 or 10 of register 8178.*

3. **Prevent programming equipment from altering register data**. Prevents the using of programming equipment to write to or change data values that are stored in all registers. *Bit 3, 7 or 11 of register 8178.*

4. **Prevent any non-programming devices from altering registers.** Prevents the use of processor-to-processor communication (priority communication) to alter data values stored in all registers. *Bit 1, 5 or 9 of register 8178.*

The bit pattern in register 8178 (the *restriction code register) determines which port (channel) is* protected, and by which restrictions. Figure 6.8 illustrates which bit in register 8178 needs to be turned ON to achieve the desired restriction and port.

To enable the restriction code, a non-zero "password number" must be entered (via the DATA ENTER mode of a programming device) into register 8177.

Once a password value is entered into register 8177, a "**-1**" is displayed when attempts are made to *monitor this register.* This keeps the password confidential. Thus, "**-1**" *cannot* itself be used as a password value.

The restriction code can be altered only by the *exact* password value entered into register 8177. When this is done, register 8177 will display "**0**".

REGISTER 8178, BIT # "ON"	*RESTRICTION AND PORT*
1	PRGMR port -- processor-to-processor communications. Restricts writing data to Model 650 registers
2	PRGMR port -- forcing of I/O is prevented using programming equipment
3	PRGMR port -- programming equipment to processor communications. Restricts writing data to Model 650 data registers.
4	PRGMR port -- restricts altering the ladder program in any way.
5	COMM port -- same effect as bit 1
6	COMM port -- same effect as bit 2
7	COMM port -- same effect as bit 3
8	COMM port -- same effect as bit 4
9	Ethernet port -- same effect as bit 1
10	Ethernet port -- same effect as bit 2
11	Ethernet port -- same effect as bit 3
12	Ethernet port -- same effect as bit 4

Figure 6.8 Restriction Code Bit Table

To *disable* the password/restriction code feature, enter a CLEAR ALL command through the PRGMR port. This is allowed regardless of the restriction code in effect, and resets the password register to zero. All rungs not protected by Inhibit Rungs are also deleted.

6.10 Memory Protect Bit

Prohibits: Ladder programming or editing through the COMM or Ethernet ports (does not affect the PRGMR port).

Uses: Control Register 8176, bit #4.

When bit 4 of register 8176 is "ON", all user ladder program, rack addressing, and labels are protected from editing. This protection feature affects the COMM and Ethernet ports (channels 2 and 3) only. It is ignored by the PRGMR port (channel 1).

To *enable* the Memory Protect Bit, two methods can be used:

1. Program a rung containing only a coil addressed as bit 4 of register 8176, like this:

2. With a programming device in the DATA ENTER mode, enter a "1" in bit 4 of register 8176.

To *disable* the Memory Protect Bit, delete the rung AND reset bit 4 to "0".

NOTE: *Merely deleting the rung does not turn bit 4 off, and by itself does not disable Memory Protect.*

6.11 Force Inhibit Bit

Prohibits: Forcing of I/O by programming equipment through the COMM or Ethernet ports (does not affect the PRGMR port).

Uses: Control Register 8176, bit #5.

When bit 5 of register 8176 is "ON" the forcing of I/O registers (with programming equipment) through the Model 650's COMM and Ethernet ports (channels 2 and 3) is prohibited. This feature is ignored by the PRGMR port (channel 1). Refer to Section 14 for detailed information on forcing.

To *enable* the Force Inhibit Bit, two methods can be used:

1. Program a rung containing only a coil addressed to bit 5 of register 8176, like this:

2. With a Programming device in the DATA ENTER mode, enter a 1 in bit 5 of register 8176.

To *disable* the Force Inhibit Bit, delete the rung AND THEN reset bit 5 to "0".

NOTE: *Merely deleting the rung does not turn bit 5 off, and by itself does not disable Force Inhibit.*

6.12 Registers Protect Bit

Prohibits: Altering of register data by any external device through the COMM or Ethernet port (does not affect the PRGMR port).

Uses: Control Register 8176, bit #6.

When bit 6 of register 8176 is ON, the altering of any Model 650 data registers through the COMM and Ethernet ports (Channels 2 and 3) is prohibited. This restriction feature is ignored by the PRGMR port.

To enable this feature, two methods can be used:

1. Program a rung containing only a coil addressed to bit 6 of register 8176:

2. Use a programming device in the DATA ENTER mode to enter a "1" in bit 6 of register 8176.

To *disable* the Register Protect Bit, delete the rung AND THEN reset bit 6 to "0".

NOTE: *Merely deleting the rung does not turn bit 6 off or disable Register Protect.*

6.13 Fenced Registers

Prohibits: Altering of any registers within a user-defined fenced area through the COMM or Ethernet ports (does not affect the PRGMR port).

Uses: Control Registers 8173 and 8174.

Registers 8174 and 8173 contain, respectively, the beginning and ending addresses of an *unprotected* user-defined block of data registers. All registers *outside* of the defined block are protected from alteration by outside devices. The default (power-up) value in register 8174 is "1"; for 8173, it's "8176". Thus, registers 8177 through 8192 are effectively fenced.

Fenced registers are alterable through the PRGMR port (Chnl 1) by other processors and programmers whose security access level is set to OVERRIDE.

7 FLOATING-POINT MATH

7.1 Introduction

This section provides information on the Model 650 floating-point (FLP) math capabilities. Floating-point math allows these Model 650 processors to perform high-level math operations on data values containing a floating decimal point.

The Model 650's floating-point operation conforms to the ANSI/IEEE Standard 754 pertaining to the 32-bit standard for floating-point math.

7.2 Definition of Terms

Compare (IF Instruction) ..... Comparison of a value in one data register to a constant, to the value of a second register, or to the result of a math operation.

Intermediate Result ... The result of one math operation in an expression containing multiple operations.

Operator ..... Math functions + − × ÷

Overflow ..... Exceeding the numerical storage range of the data register.

Result ..... The *final* value of any mathematical expression (result = A + B - C).

Soft Key ..... Also called a *function key*. Multifunction keys located above the uppermost row on the keyboard of a CRT Programmer or function keys on an IBM® keyboard using SFW Programming Software.

Transfer (LET Instruction) ..... Storing a constant numerical data value or math result in a data storage register.

7.3 Register Usage

The Model 650 processor uses different methods to deal with data stored in integer and floating-point registers. See Figure 7.1.

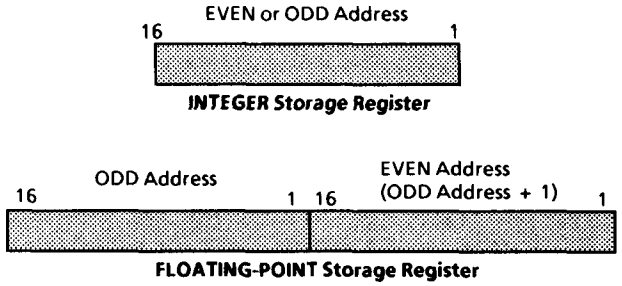


Figure 7.1 Floating Point vs. Integer Register

7.3.1 INTEGER

The range for an integer data value stored in an integer register (to be used in a math calculation) is ±32,767. The exceptions to the range limitations are registers used in timers, counters, and matrix pointers. Registers that store the current time and count can range from 0 to +9999, while matrix pointers cannot exceed 8000.

7.3.2 FLOATING-POINT OPERATIONS

The Model 650 uses two consecutively addressed 16-bit data registers to create a 32-bit floating-point data register, as shown above in Figure 7.1.

A floating-point register can contain a number from $\pm 3.4 \times 10^{38}$ to $\pm 1.2 \times 10^{-38}$. The mantissa is limited to seven digits. Values are entered and displayed using scientific notation. This notation is shown as "E" followed by an integer representing the power to which 10 is raised and multiplied by the mantissa. For example, "25,000" is displayed as "2.5 E+04". See Section 7.4.

7.4 Addressing Floating-Point Registers

Of the register pair that make up a floating-point register, *only the ODD-numbered register can be used as the address.* If the *even* address is used, an "Illegal Address" error results. The Model 650 *automatically* assigns the even-addressed register that follows the odd-addressed register. See Section 7.5 for instructions on how to directly enter floating-point numbers.

When a rung containing a floating-point register is entered into a ladder program, the letter **F** (generated by the **FLP** soft key) precedes the odd address value of that register. Refer to the following example.

EXAMPLE: *Given* - A floating-point register pair, F0051/F0052, is used to store the value 0.001234567 (1.234567×10^{-3}). The rung used to program this information is shown in Figure 7.2:

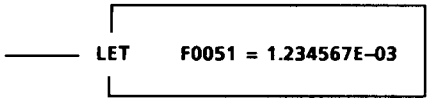


Figure 7.2 Example 1 Floating-Point LET Rung

Procedure: Press the **LET** soft key to produce another menu of soft keys.

This new menu of soft keys describes and defines the various data register types as shown below:

REG ..... Integer Register

MATRIX ..... Integer Matrix

FLP. REG ..... Floating-Point Register

FLP. MAT ..... Floating-Point Matrix

FLP. IMD ..... Floating-Point Constant

Since a floating point register is being used here, press the **FLP.REG** soft key and enter the *odd* register (0051) address. See Figure 7.3:

F0051	F0052
16 Bits	16 Bits

Figure 7.3 Example Register Structure

In Figure 7.3, F0051 specifies that register pair 0051 and 0052 will store the value 0.001234567 in 32-bit format.

Register F0052 *automatically* becomes a part of the floating-point register addressed as F0051. Once assigned as half of a floating-point register pair, register 0052 CANNOT be used for any other purpose in the same ladder program, since the bit pattern is only meaningful as part of a floating-point pair.

NOTE: *Attempting to program an even-numbered floating-point address results in an "illegal address" error.*

After entering the register address, complete the box using the "=" and "FLP.IMD" soft keys, followed by the desired floating-point number.

LADDER PROGRAM REGISTER USAGE

Any of the first 8000 user-addressable registers in the Model 650 can be designated as floating-point registers. Since each 32-bit floating-point register requires an odd-even 16-bit register pair (32 bits), 4000 floating-point registers can be used.

While it is not *necessary* to dedicate a block of registers to contain floating-point values, it is always a good idea to group together registers that serve the same purpose such as external I/O, internal I/O, integer storage, and floating-point storage.

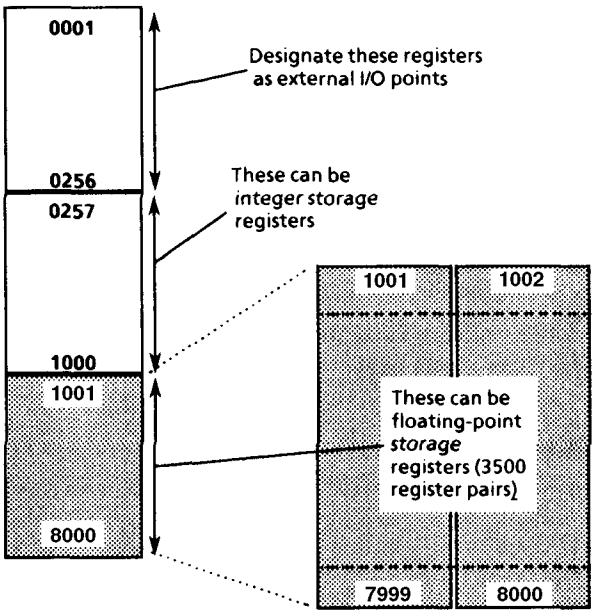


Figure 7.4 Sample Register Allocation

7.5 Data Transfers (LET) and Comparisons (IF) Using Floating-Point Numbers

Floating-point registers are allowed within two types of ladder program instructions:

1. **LET** Instructions
2. **IF** Instructions

7.5.1 DATA TRANSFERS ("LET" INSTRUCTION)

The LET operation copies a value (a constant, data value, or math result) from the right of the equal sign to the register(s) on the left side of the equal sign. Figure 7.5 shows the operation of an *integer* and a *floating point* data transfer (LET instruction).

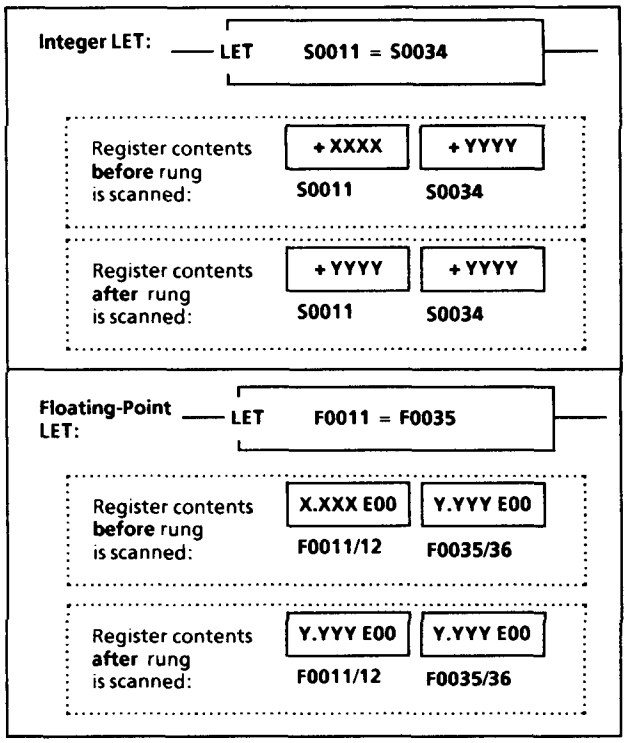


Figure 7.5 Operation of LET Instruction

See Section 7.8 for math operations allowed within LET and IF instructions.

7.5.2 COMPARISON ("IF" INSTRUCTION)

The IF instruction compares the contents of a an integer or floating-point register on the left of the COMPARE symbol to a constant or value stored in an integer or register to the right of the COMPARE symbol.

When comparing two floating-point values for equality, the two values must be *exactly* equal before the IF rung executes TRUE.

7.6 Entering Floating-Point Values

SY/MAX family devices that can enter floating-point data *directly* into floating-point registers include:

- Type SPR-300 Deluxe CRT Programmer, Series "i" (Revision 2.6 or later).
- SY/MATE Programming Software.

ENTERING FLOATING POINT NUMBERS VIA THE PROGRAMMING EQUIPMENT'S DATA MODE

To enter a floating-point value into a register, the keystrokes shown in the following table are required. Assume that the number 0.001234567 (1.234567×10^{-3}) is to be entered into odd-addressed register *0051*.

The [bracket] notation refers to a soft ("function") key, while the bold type is a dedicated keyboard key(s).

STEP #	KEY(S) USED	REMARKS
1	[DATA]	Accessed from STATUS screen.
2	[FLP.REG]	Accessed from DATA screen.
3	**51**	Enters the first address of the register pair.
4	[BIN.FLP]	When pressed, the cursor will move down one line and will show the even-numbered address (**0052**) of the selected floating-point register pair.
5	**1.234567E-3**	Enter the floating-point value
6	[ENTER]	Loads the number into the register pair.

FLOATING POINT CONSIDERATIONS

- A seven-digit mantissa coupled to a double-digit exponent is the limit of significant figure accuracy when entering a floating-point number.
- A negative value can be entered by pressing the [-] key immediately after the [BIN.FLP] soft key is pressed.
- If a value is not entered in scientific notation, the Model 650 automatically converts the value to scientific notation when the [ENTER] soft key is pressed.
- When entering a value in scientific notation, a negative exponent is specified by pressing the [-] key after the [E] key.

7.7 Displaying Floating-Point Values

When using a programming device, two methods for viewing the data stored in floating-point registers are available: the Data Display Mode and the Ladder Rung Display Mode. Each of these is explained below, and illustrated on the next page.

METHOD 1: From the DATA mode, press the FLP.REG softkey. Next, enter the register's *odd-numbered* address, then press the DISPLAY softkey (see Figure 7.6). When using the Programming Device in the DATA DISPLAY mode, floating-point values are displayed in the BIN/FLP column in the *even* (F0052) register address row.

NOTE: *The "decimal" column of the DATA mode screen displays meaningless data if the register is a floating point register.*

METHOD 2: Display the rung using the floating point register by searching for the rung, then pressing the DISPLAY soft key (see Figure 7.7).

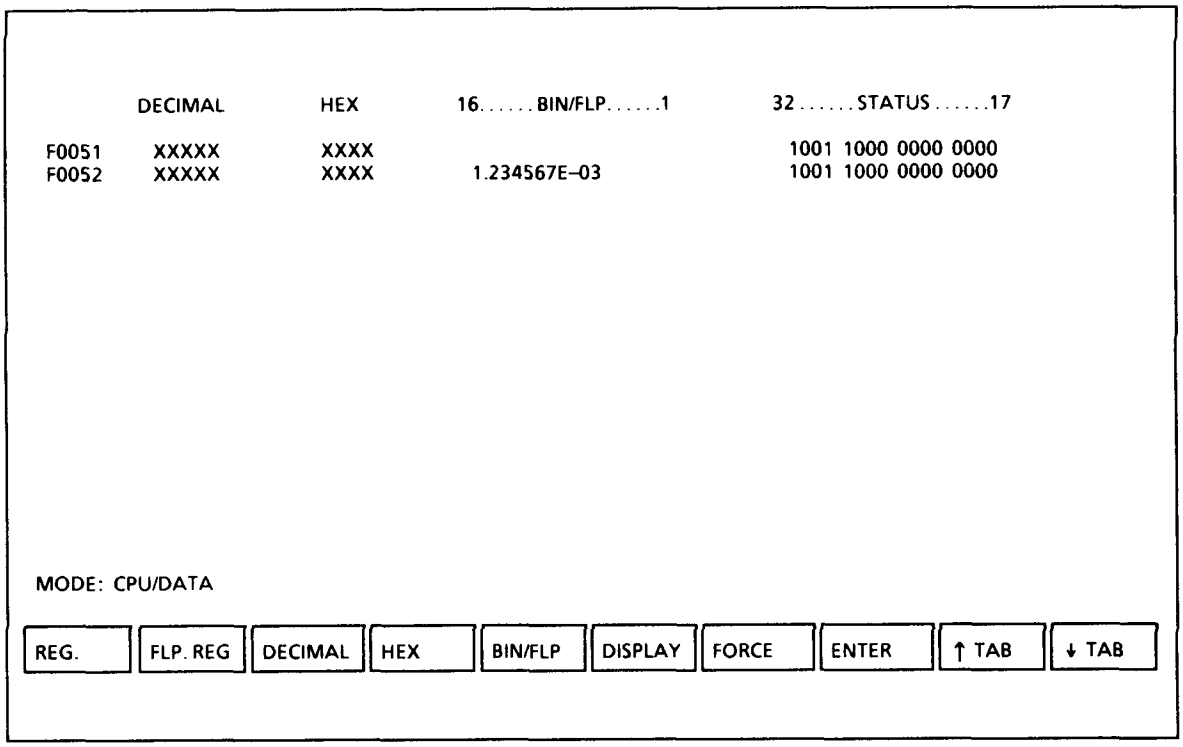


Figure 7.6 Data Display Mode

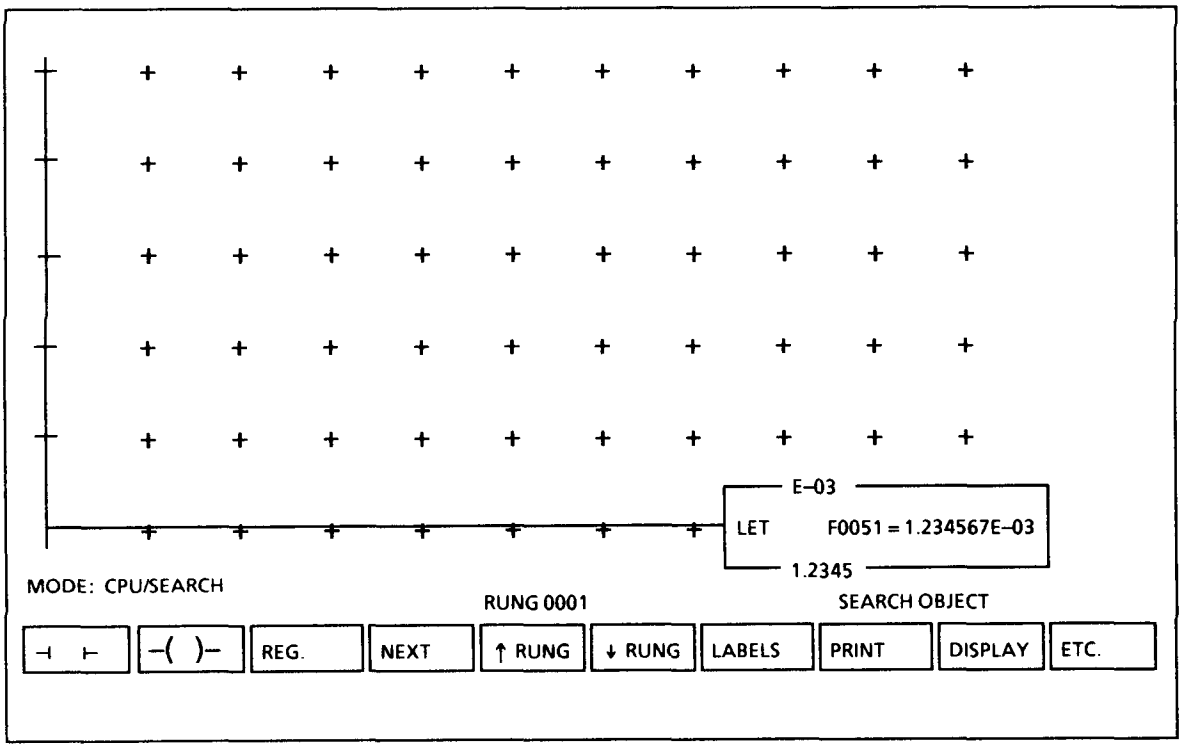


Figure 7.7 Ladder Rung Display Mode

7.8 Floating-Point Operations Within LET and IF Instructions

The Model 650 processor allows the following math functions to be programmed within "LET" and "IF" instructions.:

OPERATION	SYMBOL ON CRT
Addition	+
Subtraction	−
Multiplication	×
Division	÷
Square Root	SQRT
Sine	SIN
Cosine	COS
Common (Base 10) Logarithm	LOG
Natural (Base e) Logarithm	LN
Absolute Value	ABS
Y to the X power	Y**X

The above functions can be performed on any data values; floating point, integer, or on any constant floating point or integer.

SEQUENCE OF MATH OPERATIONS

Math operations are performed left to right within LET and IF instruction boxes and may be placed only to the right of the equal (=) or compare (<, ≥, etc.) symbol, as shown in Figure 7.8 (math operations can be placed anywhere within the dotted box).

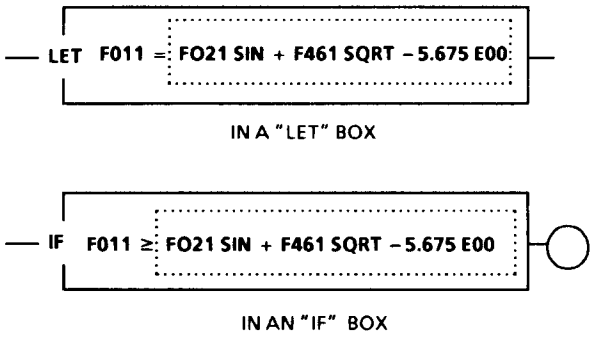


Figure 7.8 Math Operations in LET and IF Boxes

The number of math operations that can be programmed into a "LET" or "IF" rung is limited by the areas available in one row of the programming device's ladder matrix display, as shown in Figure 7.9.

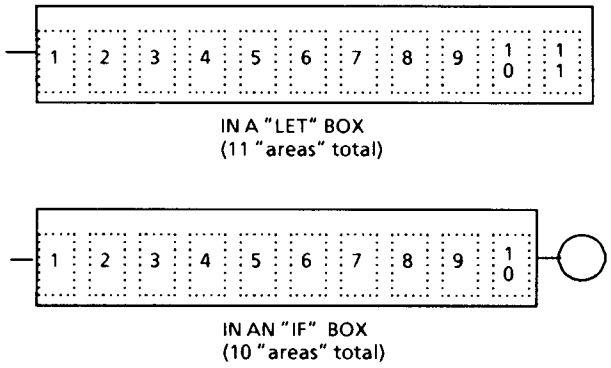


Figure 7.9 Size Limits to LET and IF Boxes

MATH PROGRAMMING CONSIDERATIONS

- The maximum number of areas available in an IF instruction is 10 when contacts are not programmed in the same rung.
- The maximum number of areas available in a LET instruction is 11 when contacts are not programmed in the same rung.

NOTE: *Multiple LET or IF rungs can be used if an instruction requires too many operations to be contained within one rung.*

7.9 Floating-Point Operation Using Matrices

In many floating-point applications, the use of matrixes, in addition to single floating-point registers, will minimize memory requirements.

A floating-point matrix or array is a group of consecutively numbered floating-point registers. The rules governing the use of matrixes are explained in the Instruction Bulletin of the programming device.

7.10 Combining Floating Point With Integer Operations

Combining floating-point and integer registers in the same LET or IF instruction is allowed in the Model 650 processor. However, the user (i.e., programmer) should be very familiar with programming techniques before attempting this type of programming option.

The following sections provide examples and considerations for combining floating point and integer math within the same LET or IF instruction.

7.10.1 LET INSTRUCTIONS

If a single LET instruction contains operations performed on both integer and floating-point values, the final result is always in the floating-point format. Refer to Figure 7.10.

The final operation stores the math result into the register specified on the left of the equal sign. If the math result (floating point or integer) to the right of the equal sign does not match the register type to the left of the equal sign, the processor converts the math result to match it. The following describes four cases in which this occurs.

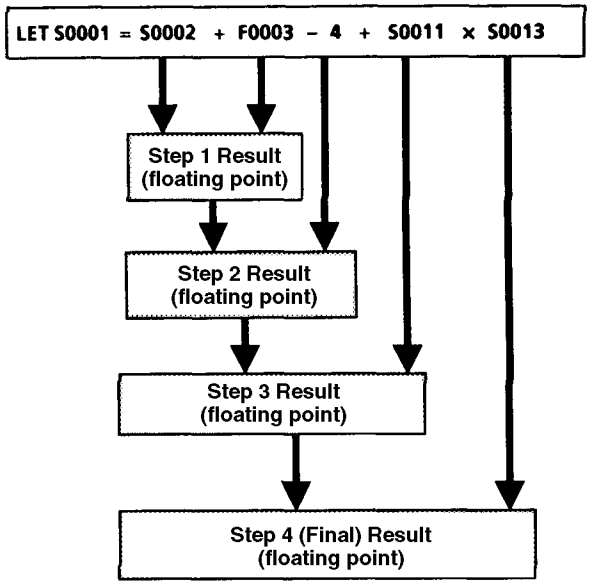
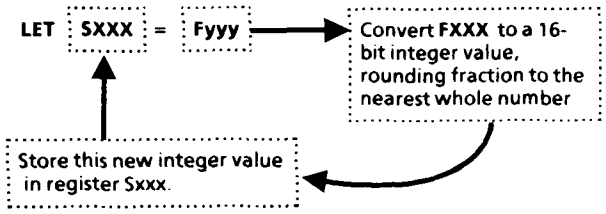


Figure 7.10 Operation Sequence in a LET Instruction Containing Both Integer and Floating-Point Registers

NOTE: *Once a floating-point value is encountered, the remaining operations are automatically converted to floating point by the processor.*

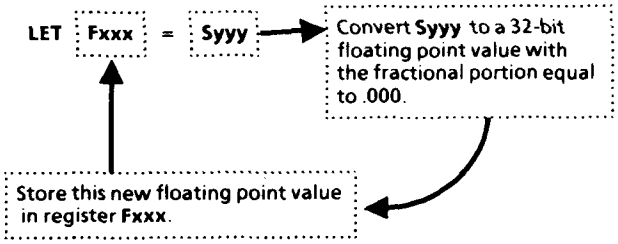
Since the processor cannot store a floating-point value in a 16-bit integer register, the floating-point value is converted to a 16-bit integer value by rounding the decimal fraction to the nearest whole number. Values less than 0.5 are rounded down, values greater than 0.5 are rounded up. For a decimal exactly equal to 0.5, the rounded result is the nearest **even** integer (for example, 2.5 is rounded down to 2.0 while 3.5 is rounded *up* to 4.0).

Case 1 - Integer Register on the Left, Floating-Point Register on the Right:

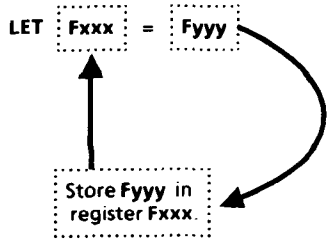


The possibility exists in Case 1 for an overflow condition. This is due to the transfer of a converted floating-point value to an integer register. Refer to Section 7.11, "Overflow Errors".

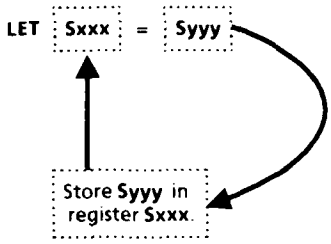
Case 2 - Floating-Point Register on Left, Integer on Right:



Case 3 - Floating-Point Register on Left, Floating-Point Result on Right:



Case 4 - Integer Register on Left, Integer Register on Right:



7.10.2 IF INSTRUCTIONS

When an IF instruction performs math operations on both integer and floating-point values, the final result on the right side of the compare will be a floating-point value. See Figure 7.11 below:

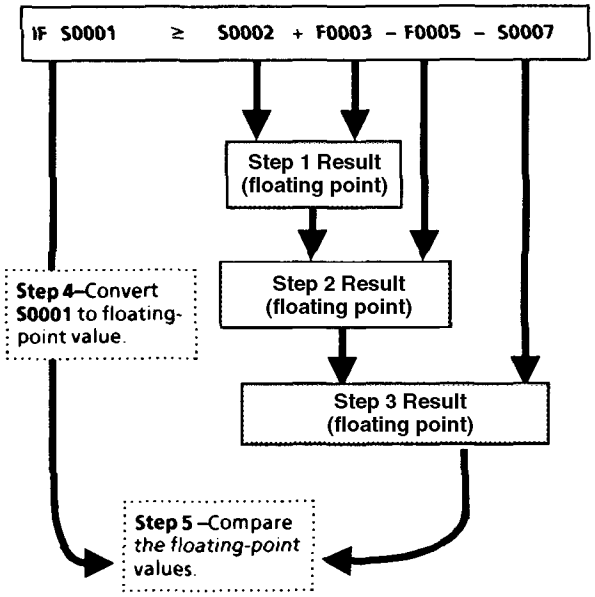
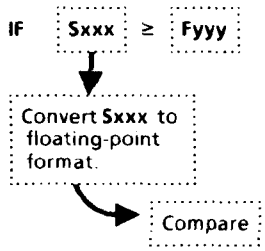


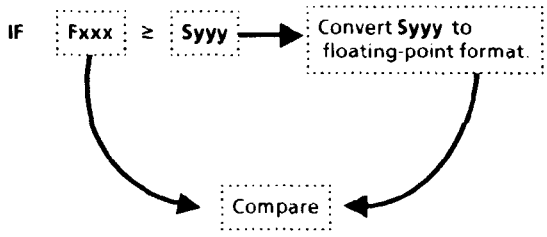
Figure 7.11 Operation Sequence in a IF Instruction That Contains Both Integer and Floating-Point Registers

If the final result of several operations is a floating-point value for comparison against an integer value, the integer value is first converted by the Model 650 to an equivalent floating-point value. The four possible "conversion/comparisons" are as follows:

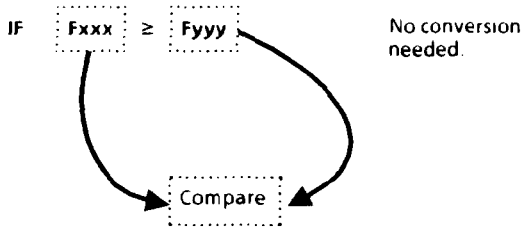
Case 1 - Compare an Integer Value to a Floating-Point Result:



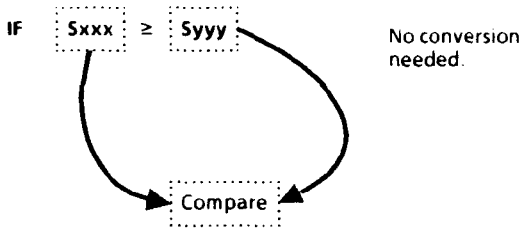
Case 2 - Compare a Floating-Point Result to an Integer Value:



Case 3 - Compare a Floating-Point Result to a Floating-Point Value:



Case 4 - Compare an Integer Value to an Integer Result:

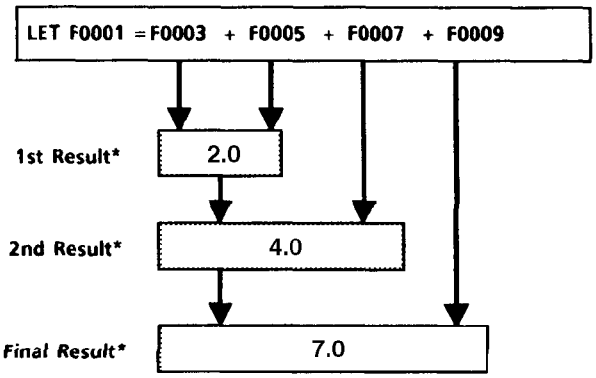


7.11 Overflow Errors

Overflow errors result from attempts to exceed a storage register's capacity.

7.11.1 ACCUMULATOR OPERATION

When LET instructions (data transfers) or IF instructions (data compares) containing math operations are scanned, the intermediate and final math results are temporarily stored in an *accumulator*. The accumulator contains the current running total of the math operation and is shown as a shaded rectangle in Figure 7.12 below:



*Given that F0003 = 1.0, F0005 = 1.0, F0007 = 2.0, F0009 = 3.0

Figure 7.12 Accumulator Operation

The accumulator can store a much larger value than floating point or integer storage registers. Numbers in the accumulator can range from $\pm 3.4 \times 10^{-4932}$ to $\pm 1.2 \times 10^{+4932}$.

NOTE: *The operator has no direct control over the operation or contents of the accumulator unless one of the "Alternate Accumulator Manipulations" functions is used. Refer to Section 8.5 and pages 8-2 and 8-3.*

7.11.2 WHY DO OVERFLOWS OCCUR?

Overflow errors are generated any time an attempt is made to exceed a storage register's capacity. The following table shows register types and allowable storage ranges. Figure 7.14 illustrates where overflows can occur.

REGISTER TYPE	CAPACITY
Integer	± 32,767*
Integer Accumulator	± 2,147,483,647*
Floating Point	$\pm 1.2 \times 10^{-38}$ to $\pm 3.4 \times 10^{38}$
Floating-Point Accumulator	$\pm 1.2 \times 10^{+4932}$ to $\pm 3.4 \times 10^{-4932}$

* *Using -32,768 or -2,147,483,648 in a math statement may produce an overflow error.*

Although the Model 650's control processor is designed to prevent the floating-point math coprocessor from generating a result of infinity from a calculation (caught by the overflow checking), infinity may be introduced into a floating-point register by loading the values shown in Figure 7.13.

REGISTER	Indication of Exceeding Positive Overrange Limit ($+3.4 \times 10^{38}$)	
	HEX	BIN/FLP
F0051	0000	None
F0052	7F80	+INFF

REGISTER	Indication of Exceeding Negative Overrange Limit (-3.4×10^{38})	
	HEX	BIN/FLP
F0051	0000	None
F0052	FF80	-INFF

Figure 7.13 Overflow Indicators

CAUTION

Avoid entering any infinity-producing value into a floating-point register. Infinity is a mathematically invalid result that defeats the purpose of the overflow bits, causing these bits to be unreliable. Once present, infinity may propagate throughout other calculations and produce unwanted results.

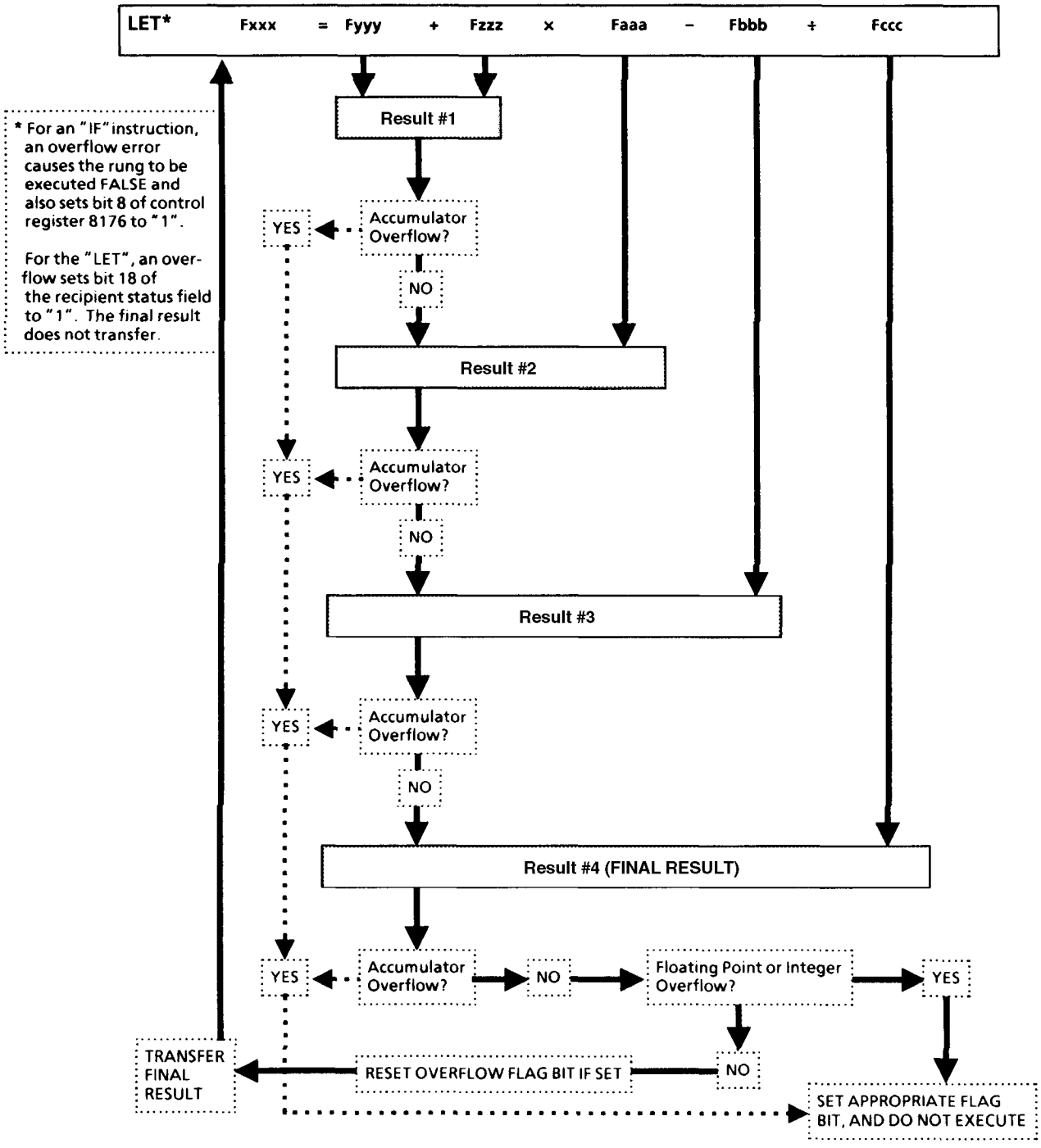


Figure 7.14 Overflow Error Points

7.12 Special Math Functions

SY/MAX programming devices such as the SPR-250, SPR-300, and SYM Programming Software use the symbols shown in the following table to represent some of the special math functions.

CRT SYMBOL	MATH FUNCTION
SIN	Sine
COS	Cosine
SQRT	Square Root
LOG	Common (base 10) Logarithm
LN	Natural (base e) Logarithm
Y**X	Raise Y to the power of X
ABS	Absolute Value

The special function always operates on the intermediate math result created by any math expressions preceding it (refer to Figure 7.15).

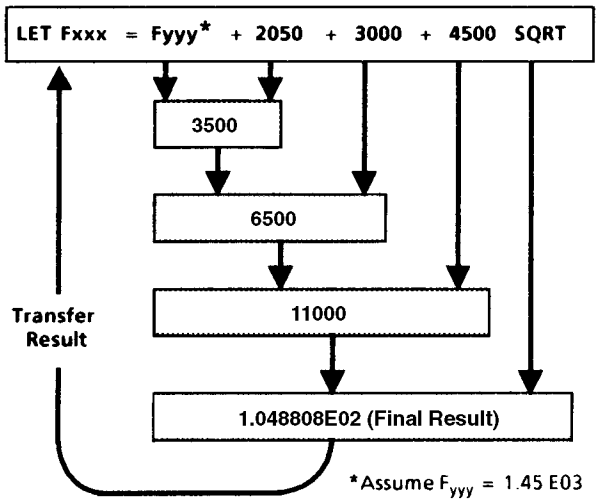


Figure 7.15 Special Function Operation

RESTRICTIONS

The value that a math function operates on is called the *argument*. Certain restrictions apply regarding the sign and magnitude of the argument.

Figure 7.16 illustrates the allowable argument conditions for some of the special math functions.

Special Function	ARGUMENT CONDITION ("Y" indicates argument *is* allowed, "N" means argument *not* allowed)					
	Zero	Final Result Over FP Range	Value −	Value +	Inter-med. Result Over FP Range	÷ By 0
SQRT in LET or IF	Y	N	N	Y	Y	N
SIN/COS in LET or IF	Y	N	Y	Y	Y	N
LOG/LN in LET or IF	N	N	Y	Y	Y	N
Y**X (Y) in LET or IF	Y	N	N	Y	Y	N
Y**X (X) in LET or IF	Y	N	Y	Y	Y	N
ABS in LET or IF	Y	N	Y	Y	Y	N

Figure 7.16 Allowable Special Function Arguments

NOTE: *Due to the behavior of the math coprocessor, small inaccuracies may be observed at function range extremes and points of discontinuity. See the following.*

TRIGONOMETRIC FUNCTION DISCONTINUITIES

The following results may be obtained when nearing points of discontinuity in calculations:

- The sine of 0, 180 and 360° is 0; the Model 650, however, may yield very small non-zero results.
- The tangent of 90° is infinity; the Model 650 may report it to be an extremely *large* number (–3.7E19, typically).
- The tangent of 180° is 0; the Model 650 may show it to be an extremely *small* number.

Results similar to the above could occur when using any trig function. If a problem is anticipated, it should be compensated for in the user program.

8 SPECIAL INSTRUCTIONS

8.1 Description

The Model 650 instruction set has been expanded beyond the use of floating-point five-function math, sine, cosine, logarithms (base 10 and natural), absolute value, and exponential math. Refer to Section 7 for a discussion of floating-point math and the utilization of the listed functions.

NOTE: *It is strongly recommended that the concepts in Section 7 be understood before proceeding with this section.*

Programming the following instructions is accomplished by pressing the SPECIAL soft key, accessible when constructing an IF or LET rung, followed by the corresponding function number associated with the instruction. The appropriate function is performed on the intermediate mathematical result that exists just prior to encountering the SPECIAL instruction.

NOTE: *All mathematical operations are performed from left to right and each operation is completed before the next operation in the rung is executed.*

Certain functions, by definition, require a *double argument*. The user should program a semicolon (from the soft key display) after the function number, followed by the second argument (which may be either a constant or a register value).

Programming any of the single argument functions results in the indicated operation being performed on the intermediate result. To program, the user must begin constructing a rung using the IF or LET box format. The desired operation can then be applied to any intermediate result which exists to the right of the "=" sign.

To implement the desired function, begin with the following "starting point" soft key display:

REG	MATRIX	FLP REG	FLP MAT	FLP IMD	+	-	X	÷	ETC

Striking the ETC soft key produces a new display which looks like the following:

AND	OR	XOR	SPECIAL	SIZE		;			ETC

Striking the SPECIAL soft key produces the following display:

SIN	BCD	BIN	COS	SQRT	LN	LOG	ABS	Y**X	

The user may now strike the proper soft key if one of the listed functions is to be used, or may instead enter from the keyboard the numeric value which corresponds to one of the additional SPECIAL functions. In the latter case, the entered number appears in the IF or LET rung box and the soft key display reverts to the "starting point" soft key display.

After entering the desired numeric value, the user may either exit the box (if finished constructing the mathematical expression) or continue building the expression by following the SPECIAL instruction with either another SPECIAL instruction or a mathematical operator (+, –, X, ÷). In either case, the abbreviation SPEC appears in the box to indicate the appropriate SPECIAL function has been entered in the rung.

Programming the double argument functions is similar except that the user has to provide the second argument in order for the function to be programmed correctly. This must be provided IMMEDIATELY after the numeric value which identifies the SPECIAL double argument function. Thus, the keystrokes employed for the double argument functions are the same as for the single argument function, but require the following additional keystrokes:

1. After entering the numeric value which corresponds to the desired SPECIAL function, the soft key display reverts to the "starting point" display:

REG	MATRIX	FLP REG	FLP MAT	FLP IMD	+	-	X	÷	ETC

2. Striking the ETC soft key changes the display to the following:

AND	OR	XOR	SPECIAL	SIZE		;			ETC

3. Striking the semicolon (;) soft key, followed by the ETC key (the display does not change after striking the ";" soft key) prepares the box for the second argument. The following is displayed:

REG	MATRIX	FLP REG	FLP MAT	FLP IMD	+	-	X	÷	ETC

The second argument can now be entered as either a constant (integer or floating point number) or a variable (integer register or floating point register). Upon entering the second argument, the user may either exit the box (if finished constructing the mathematical expression) or continue building the expression by entering either another SPECIAL instruction or a mathematical operator (+, –, X, ÷). In either case, the abbreviation "SPEC" appears in the box to indicate the appropriate SPECIAL instruction has been entered in the rung.

PROGRAMMING CONSIDERATIONS

- The first element to the right of the "=" sign cannot be a mathematical operator (+,–, X, or ÷) or a SPECIAL function.
- Entering a numeric value for a SPECIAL function that does not exist (i.e., a "Special Number" greater than 65) generates PROCESSOR ERROR 5 when the rung is loaded.
- When programming SPECIAL functions with two arguments, the *EXACT* order of entry is critical. The numeric value for the function *must be followed immediately by the semicolon from the softkey display, which in turn must be followed immediately by the second argument.*

Along with the terminology in Section 7.2, familiarity with the following terms is necessary to understand the rest of this section:

Floating Point Accumulator ..... An 80-bit "scratch-pad" register which contains the intermediate result of the Model 650's floating point calculations. It provides precision that would normally be lost when using a normal 32-bit floating-point register. Also discussed in Sections 7.11 and 8.5.

Integer Accumulator .. A 32-bit register which contains the intermediate result of an integer operation.

Alternate Accumulator A different 80-bit register that can be accessed by the user to exchange or swap information in the floating-point accumulator.

Intermediate Result (IR) The data register immediately to the right of the "=" sign, or the current running total of a computation. The IR is stored in the accumulator.

New Intermediate Result (NIR) ..... The result obtained after applying a SPECIAL function to the intermediate result.

ARG2 ..... The notation for the second argument of a double-argument function.

ALT ..... Notation for the contents of the alternate accumulator.

Figure 8.1 lists all the special (SPEC) instructions the Model 650 processor can perform. The SPEC portion of the rung is shown below:

Both types of the Model 650 can perform all of the listed instructions.The sample rungs used in Figure 8.1 are for illustrative purposes only and are not intended to produce practical results. Most of the registers shown are integer type registers. For certain functions (in particular the trig functions), a *floating point* register would produce more accurate results. Also, IF statements could have been used in place of the LETs in most cases.

SPEC #	Function	Action	Example
0	SIN(X)	Calculate sine of x	SPEC LET F11 = F21 SIN
1	BIN to BCD	Convert binary pattern to Binary Coded Decimal	SPEC LET S10 = S20 BCD
2	BCD to BIN	Convert BCD value to binary	SPEC LET S10 = S20 BIN
3	COS(X)	Calculate cosine of x	SPEC LET F11 = F21 COS
4	SQRT	Calculate square root of a positive value	SPEC LET S10 = S20 SQR
5	LN(X)	Calculate natural log of x	SPEC LET F11 = F21 LN
6	LOG(X)	Calculate common log of x	SPEC LET F11 = F21 LOG
7	|X|	Calculate absolute value of x	SPEC LET S10 = S20 ABS
8	Y**X	Raise Y to the *power of X*	SPEC LET S10 = S20 Y**X;S30
9	Y**ALT	Raise Y to the power of the value stored in the alternate accumulator	SPEC LET S10 = S20 9
10	1/X	Calculate the reciprocal of x	SPEC LET F11 = F21 10
11	SWAP 16	High byte and low byte of IR are exchanged	SPEC LET S10 = S20 11
12	*SWAP 32*	Most significant word in the IR *becomes the NIR*	SPEC LET S10 = S20 12
13	READ STAT	Indirect register read of status field (data in S20 points to register where status field is found)	SPEC LET S10 = S20 13
14	READ DATA	Indirect register read of data field (data in S20 points to the register where data field is located)	SPEC LET S10 = S20 14

Figure 8.1 Special Instruction List (1 of 5)

SPEC #	Function	Action	Example
15	FIND BIT	Indirect register bit search (find and reset the lowest bit set in S20. Store 1 to 16 to identify bit, or store -32,768 if all bits are at "0")	LET S10 = S20 SPEC 15
16	NEGATE	Change sign of current number	LET S10 = S20 SPEC 16
17	NUM TYPE	Returns type of current number	LET S10 = S20 SPEC 17
18	ACC ALT	Exchange current and alternate accum-ulators	LET S10 = S20 SPEC 18
19	ADD ALT	Add S20 to alternate accum-ulator	LET S10 = S20 SPEC 19
20	SUB ALT	Subtract S20 from alternate accum-ulator	LET S10 = S20 SPEC 20
21	DUP ALT	Copy alternate accumulator into current accum-ulator	LET S10 = S20 SPEC 21
22	DUP ACC	Copy current accumulator into alternate accum-ulator	LET S10 = S20 SPEC 22
23	RAN-DOM	Generates a random number (from 0 to value of current accum-ulator minus 1)	LET S10 = S20 SPEC 23
24	ROUND	Rounds FP numbers to integers	LET S10 = F21 SPEC 24
25	SQUARE	Squares current result	LET S10 = S20 SPEC 25
26	HYPOT	Calculate two-dimensional hypotenuse $\sqrt{IR^2 + ARG2^2}$	LET F11 = F21 SPEC 26;F31
27	HYPALT	Calculate two-dimensional hypotenuse $\sqrt{IR^2 + ALT^2}$	LET F11 = F21 SPEC 27

Figure 8.1 Special Instruction List (2 of 5)

SPEC #	Function	Action	Example
28	HYP XYZ	Calculate three-dimensional hypotenuse $\sqrt{IR^2 + ARG2^2 + ALT^2}$	LET F11 = F21 SPEC 28;F31
29	ADD STAT	Perform statistical summation S20 = IR = data, S30 = ARG2 = pointer	LET S10 = S20 SPEC 29;S30
30	VARI-ANCE	Calculate variance on statistical summation	LET F11 = S30 SPEC 30
31	TAN(X)	Calculate tangent of x	LET F11 = F21 SPEC 31
32	COSEC (X)	Calculate 1/sin(x)	LET F11 = F21 SPEC 32
33	SECANT (X)	Calculate 1/cos(x)	LET F11 = F21 SPEC 33
34	COTAN (X)	Calculate 1/tan(x)	LET F11 = F21 SPEC 34
35	ARCSIN (X)	Calculate $\sin^{-1}(x)$	LET F11 = F21 SPEC 35
36	ARCCOS (X)	Calculate $\cos^{-1}(x)$	LET F11 = F21 SPEC 36
37	ARCTAN (X)	Calculate $\tan^{-1}(x)$	LET F11 = F21 SPEC 37
38	ARCTAN (Y/X)	Calculate $\tan^{-1}(y/x)$	LET F11 = F21 SPEC 38;F31
39	ARCCSC (X)	Calculate $\sin^{-1}(1/x)$	LET F11 = F21 SPEC 39
40	ARCSEC (X)	Calculate $\cos^{-1}(1/x)$	LET F11 = F21 SPEC 40
41	ARCCOT (X)	Calculate $\tan^{-1}(1/x)$	LET F11 = F21 SPEC 41
42	SET DEGREE	Set degree mode for trig calculations	LET F11 = F21 SPEC 42

Figure 8.1 Special Instruction List (3 of 5)

SPEC #	Function	Action	Example
43	SET RADIAN	Set radian mode for trig calculations	LET F11 = F21 SPEC 43
44	DEG RAD	Convert degrees to radians	LET F11 = F21 SPEC 44
45	RAD DEG	Convert radians to degrees	LET F11 = F21 SPEC 45
46	PID REV	Calculate reverse-acting PID loop	LET S517 = 51 SPEC 46
47	PID DIR	Calculate direct-acting PID loop	LET S517 = 51 SPEC 47
48	PID MAN	Calculate manual mode for PID loop	LET S517 = 51 SPEC 48
49	LEAD /LAG	Perform the lead/lag function on a process variable	LET S56 = 71 SPEC 49
50	PI	Multiply by pi (3.1415....)	LET F11 = F21 SPEC 50
51	L2T	Multiply by log base 2 of 10	LET F11 = F21 SPEC 51
52	L2E	Multiply by log base 2 of e	LET F11 = F21 SPEC 52
53	LG2	Multiply by log base 10 of 2	LET F11 = F21 SPEC 53
54	LN2	Multiply by log base e of 2	LET F11 = F21 SPEC 54
55	EULER	Multiply by Euler's Constant	LET F11 = F21 SPEC 55
56	E	Multiply by e (natural logarithm)	LET F11 = F21 SPEC 56
57	1 DEG	Multiply by one degree (in radians)	LET F11 = F21 SPEC 57
58	1 RAD	Multiply by one radian (in degrees)	LET F11 = F21 SPEC 58
59	E**PI	Multiply by e raised to the power of pi	LET F11 = F21 SPEC 59

Figure 8.1 Special Instruction List (4 of 5)

SPEC #	Function	Action	Example
60	PI**E	Multiply by *pi* raised to the power of e	LET F11 = F21 SPEC 60
61	E**E	Multiply by e raised to the power e	LET F11 = F21 SPEC 61
62	E** EULER	Multiply by e raised to the power of Euler's Constant	LET F11 = F21 SPEC 62
63	SQR(E)	Multiply by the square root of e	LET F11 = F21 SPEC 63
64	SQR(PI)	Multiply by the square root of *pi*	LET F11 = F21 SPEC 64
65	DRUM	Initiates Drum Sequencer operation (the second argument is the number of steps in the sequence. See section 11).	LET S21 = S21 SPEC 65;20
66	-	Reserved for future use.	
67	BWRITE	Enables backplane write to Microcell (the second argument is the register count, see Section 8.6).	LET S1 = S1001 SPEC 67;28

Figure 8.1 Special Instruction List (5 of 5)

8.2 Math and Trig Operations

The operations listed in Figure 8.2 are designed to enhance the functions listed in the programmer's Instruction Bulletin. A number of high-precision constants are available in the Model 650, along with the ability to perform trig operations in either degree or radian mode.

APPLICATION CONSIDERATIONS

- The default mode for trig operations is *degrees* (initially set upon a HALT-to-RUN transition).
- The selected mode remains in effect until either a SPEC 42 or SPEC 43 is encountered or until a HALT-to-RUN transition occurs.

Instruction	SPEC #	Description
INVERT	10	Takes the reciprocal of the intermediate result (NIR = 1/IR)
NEGATE	16	Changes the sign of the intermediate result (NIR = 0–IR)
ROUND	24	Rounds the current intermediate result to the nearest integer number. The value 0.5 is converted to the nearest *even* integer.
XSQRD	25	Squares the intermediate result
HYPOT	26	Calculates the two-dimensional hypotenuse This is a *double-argument* function.
HYPXYZ	28	Calculates the three-dimensional hypotenuse, a *double-argument* function.
SETDEG	42	All subsequent calculations will be performed in *degrees*
SETRAD	43	All subsequent calculations will be performed in *radians*
DEGRAD	44	Converts degrees to radians
RADDEG	45	Converts radians to degrees
TANGNT	31	Calculates the tangent
COSEC	32	Calculates the cosecant
SECANT	33	Calculates the secant
COTAN	34	Calculates the cotangent
ARCSIN	35	Calculates the arc sine
ARCCOS	36	Calculates the arc cosine
ARCTAN	37	Calculates the arc tangent
ARCCSC	39	Calculates the arc cosecant
ARCSEC	40	Calculates the arc secant
ARCCOT	41	Calculates the arc cotangent

Figure 8.2 Special Trig/Math Operations

- Although the CRT is limited to displaying seven figures, the internal accumulator precision remains at 80 bits and the floating-point register's precision is 32 bits.
- All other rules for IF and LET functions apply.

In Figure 8.2 the second column shows the "Special Number" which is used in the LET box to generate that function. The SPEC number is the same that appears in the first column of the preceding table.

TRIG AND MATH CONSTANTS

Figure 8.3 below gives a list of constants available for use in the Model 650 programming set. Entering the numeric function results in the intermediate result being multiplied by the indicated constant, which is stored to 80-bit (18 significant digit) precision.

Instruction	SPEC #	IR Will be Multiplied by:
PI	50	Pi to 18 significant digits
L2T	51	Log base 2 of 10
L2E	52	Log base 2 of the value of e
LG2	53	Log base 10 of 2
LN2	54	Log base e of 2
EULER	55	Euler's Constant
E	56	Natural log e
1DEG	57	One degree, in radians
1RAD	58	One radian, in degrees
ETOPI	59	e raised to the power of *pi*
PITOE	60	*pi* raised to the power of e
ETOE	61	e raised to the power of e
ETOEULR	62	e raised to the power of Euler's Constant
SQRTE	63	The square root of e
SQRTPI	64	The square root of *pi*

Figure 8.3 Special Constants

NOTE: *The new intermediate result in the following table is denoted as "N".*

Instruction	SPEC #	Statistical Operation
ADDSTA	29	Adds the value preceding the SPEC 29 entry to the statistical block of registers (defined by the pointer), as follows: **F1 = SUM(X) = ΣX** **F3 = SUM(X^2) = ΣX^2** **S5 = N, where N is the number of terms which have been summed.** To be of practical value, the LET statement that initiates the ADDSTA function must be *transition-sensitive*. If it isn't, the summation operations will be performed every scan. The maximum legal value for N is +32,767.
VARNCE	30	Calculate the variance on the accumulated statistical block. The number preceding the SPEC 30 in the LET box must point to the same block of registers used by the ADDSTA function. $NIR = \frac{N*[SUM(X^2)] - [SUM(X)]^2}{N*(N-1)}$ $= \frac{N * \Sigma X^2 - (\Sigma X)^2}{N*(N-1)}$ Since the *standard deviation* is, by definition, the square root of the variance, the standard deviation can also be obtained by taking the square root of the result of the VARNCE operation. In the same way, the *mean* can be obtained by dividing the first floating point register assigned to the statistical block by the fifth, as in the following: $MEAN = \frac{SUM(X)}{N} = \frac{\Sigma X}{N} = \frac{F1}{S5}$

Figure 8.4 Statistical Functions

8.3 Statistical Operations

The number preceding SPECIAL 29 in the LET box (X) represents the numeric quantity to be operated on and then stored in the statistical block. The number can be contained in either an integer or floating-point register.

The second argument (which follows the semicolon) may be a constant or a variable and must be an *odd integer ranging from 1 to 7995*. This integer is used as a pointer to a user-specified block of 5 registers of accumulated data. Note that the first four registers are actually a pair of floating-point registers and the fifth is an integer register.

Instruction	SPEC #	Read Operation
RDSTAT	13	Reads the *status* field of the *indicated indirect register*. The status field becomes the new IR (NIR = status field of the register pointed at by the *IR*).
RDDATA	14	Reads the *data* field of the indicated indirect register. The data field becomes the new IR (NIR = data field of the register pointed at by the IR).
FNDBIT	15	Locates and identifies the lowest bit set in the indirect register data field. The lowest bit position that is set becomes the new IR, then the bit itself is reset. If no bits are found to be set, the NIR becomes 8000H (-32,768). Thus, the only values which the NIR can become after execution of FNDBIT are 1 to 16 (if bit IS found), or -32,768 (if bit ISN'T found). To be of practical value, the LET statement that initiates the FNDBIT function should be *transition sensitive*. If not, the operation is performed every scan and the user may not perceive proper operation.

Figure 8.5 Register READ Operations

8.4 Indirect Register Read Operations

For the functions listed in Figure 8.5, the number preceding the SPEC function in the LET box acts as an indirect pointer to the desired register. The indirect pointer must be an integer (representing a number or a storage register) ranging from 1 to 8000 or 8097 to 8192.

Instruction	SPEC #	Manipulation Performed
YTOALT	9	Similar to the math operation Y^x, except that Y is raised to the power of the alternate accumulator (NIR = IR^{ALT})
ACCALT	18	Exchanges the intermediate result with the alternate accumulator (IR becomes ALT and ALT becomes IR)
ADDALT	19	Adds the intermediate result to the alternate accumulator. The intermediate result remains the same.
SUBALT	20	Subtracts the intermediate result from the alternate accumulator. The intermediate result remains the same.
DUPALT	21	Duplicates the alternate accumulator. The NIR assumes the value in the alternate accumulator (NIR = ALT), and the alternate accumulator remains unchanged.
DUPACC	22	Duplicates the accumulator (intermediate result). The intermediate result is copied into the alternate accumulator (ALT = NIR), and the IR remains unchanged.
HYPALT	27	Calculates the hypotenuse based on results stored in the alternate and IR accumulators $\lvert NIR = (\sqrt{IR^2 + ALT^2}) \rvert$
ATANYX	38	Calculates the arc tangent for the ratio of the intermediate result to the second argument [NIR = ARCTAN (IR/ARG2)]

Figure 8.6 Accumulator Manipulations

8.5 Alternate Accumulator Manipulations

The functions in Figure 8.6 allow access to the two 80-bit floating-point accumulators in the Model 650. These accumulators maintain a high accuracy which might otherwise be lost in a standard 32-bit floating point register. See also Section 7.11 and pages 8-2 and 8-3.

8.6 Bus Write to Microcell

Instruction	SPEC #	Action Performed
BWRITE	67	Performs an immediate write of data to the specified register, with the count value indicated by the second argument.

Figure 8.7 Bus Write to Microcell

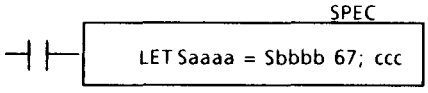
When exchanging data with the Microcell via the backplane, the image table operation of the Model 650 processor requires some special considerations not explicitly stated in Section 8 or Section 11.4 of the Microcell Instruction Bulletin #30598-275-xx. The following technique for reading and writing registers applies to **Model 650 processors, Revision 2.00 or later**, used with Microcells which are Revision 4.0 and later. Earlier revision Microcells will allow Model 650 processors to write data over the backplane, but not to read any registers.

Reading Registers

All rack addressed Microcell registers (maximum of 256) are read by the processor into its image table as part of normal End-of-Scan updating.

Writing Registers

Any rack addressed Microcell registers can be written to the processor by utilizing the following SPECIAL command.



where: aaaa = first Microcell destination register.
bbbb = processor source register
ccc = register count

SPECIAL 67 is similar to an Immediate I/O Update instruction and results in the contents of the block of processor registers starting at address bbbb to be written to the block of Microcell registers starting at address aaaa, with the block size determined by the register count ccc. Note this is different from a standard LET statement which only transfers data

within the image table registers of the processor. The following application considerations apply:

- The processor must be Rev. 2.00 or greater; the Microcell must be Rev. 4.0 or greater.

- The format must be exactly as shown above; bit 18 in the status field of reg. aaaa will be set to indicate the instruction has failed to execute due to improper formatting.

- The Microcell must be located in the same rack as the processor.
- A minimum of 28 registers (maximum of 256) should be rack addressed to the Microcell.

- The SPECIAL 67 instruction, Figure 8.7, is intended to be used only for performing direct bus updates of Microcell registers, and should not be applied to other modules (use Immediate I/O Update instruction instead).

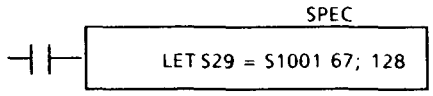
The following examples illustrate the use of the SPECIAL WRITE command to update registers in the Microcell. Note any of these registers can be READ using standard (e.g. IF) instructions.

Example 1 – Updating the FILE COMMAND Registers

Section 11.4 of the Microcell Instruction Bulletin #30598-275-xx shows three examples for downloading a file, appending an MCM file, and shutting down the MCM by writing to the first 28 Microcell registers. Each example uses a ladder rung employing a MATRIX statement to accomplish the register data transfers. Each of these statements needs to be replaced by the SPECIAL 67 format, with the "PNTR" value being replaced by the "count" value of the new instruction.
Thus, the rung of section 11.4.1 now becomes

Example 2 – Reading and Writing the BUS DATA Registers

Assume the first 156 system registers have been rack addressed to the Microcell. Registers 29-156 are the BUS DATA registers, available for reading and writing register values. Because these Microcell registers are automatically read into the processor image table at the end of each scan, they can be monitored with the standard instruction set.
In order to write the contents of (for example) registers 1001-1128 into these registers, use the SPECIAL instruction.



Note if a regular LET or MATRIX statement was used instead to update registers 29 through 156, only the processor's internal image table registers would be updated; these registers would subsequently be overwritten when the corresponding Microcell registers were automatically read during normal End-of-Scan updating.

8.7 Miscellaneous Instructions

The table in Figure 8.7 lists the various miscellaneous instructions that are available with the Model 650 processor.

Instruction	SPEC #	Action Performed
RANDOM	23	Generates a random number. The number preceding the SPEC 23 box is used as the argument value for the generator. The new IR will fall in the range 0 to (IR–1). **NOTE:** *An argument value of 0 or 1 will return a random number value of 0, while an argument greater than 32,767 or less than 0 produces an error.*
SWAP 16	11	Exchanges the high byte (9-16) and low byte (1-8) of the IR. The IR must be an integer in the range ± 32,767.
SWAP 32 (Integers only)	12	Copies the high word (17-32) of the IR to the low word (1-16) of the IR. The high word is sign extended, either all "0"s (if bit 16 is a "0"), or all "1"s (if bit 16 is a "1").
NUMTYP	17	Identifies the number type of the intermediate result. The IR becomes a value from 0 to 14, according to standards applied to the Motorola 68010 microprocessor and 68881 math coprocessor as shown below: 0 Positive normalized or de-normalized 1 Positive not-a-number 2 Positive infinity 3 *Unused* 4 Positive zero 5-7 *Unused* 8 Negative normalized or de-normalized 9 Negative not-a-number 10 Negative infinity 11 Integer zero 12 Negative zero 13 Negative integer 14 Positive integer

Figure 8.7 Miscellaneous Operations

9 PID OPERATIONS

9.1 Introduction

NOTE: *A more complete discussion of the equations and terminologies used in this section can be found in Instruction Bulletin 30598-301-0X ("PID Closed Loop Control") or 30598-240-0X ("Process Control Module").*

Algorithms are available in the Model 650 processor for solving the following three types of PID loops:

1. Reverse-acting (PIDR)
2. Direct-acting (PIDD)
3. Manual (PIDM)

The number that precedes SPEC 46, 47 or 48 in the Special Instruction LET box (an odd integer from 1 to 7981) is a pointer to a user-defined block of 18 storage registers.

9.2 Register Allocation

The block of 18 storage registers provides information needed by the Model 650 to execute the PID algorithms, and must follow the format shown in Figure 9.1.

The "L" prefix in Figure 9.1 indicates the relative position within the 18-register block. For example, if registers 51 through 68 are set aside for the PID algorithm registers, "L1" would refer to register 51, "L5" would refer to register 55, and so on. The boldface items in Figure 9.1 are parameters used in equations.

REGISTER #	PURPOSE
L1, L2	Integral Summation [**SUM (I)**]. These two registers must be a floating-point register.
L3	Value of the process variable used the last time the loop was updated (**PVn–1**)
L4	Not Used
L5	The output from the loop solution (**OUTPUT**)
L6	Value of the process variable used during the present loop update (**PVn**)
L7	The set point (**SP**)
L8	LOOP STATUS *BIT 2*: 1 if loop *is not* in manual 0 if loop *is* in manual *BIT 3*: 1 if loop *is* in automatic 0 if loop *is not* in automatic *BIT 10*: 1 if loop is *direct-acting* 0 if loop is *reverse-acting* *BIT 14*: 1 if high output limit exceeded 0 if high output limit *not* exceeded *BIT 15*: 1 if low output limit exceeded 0 if low output limit *not exceeded* *BIT 16*: 1 if low output limit is illegally set to be higher than the high limit
L9, L10	Not Used
L11	Proportional (gain) constant (K_p) This is a dimensionless number.
L12	Integral (reset) constant (K_I) in repeats per minute
L13	Derivative (rate) constant (K_D) in seconds
L14	Bias or offset (**B**) in %
L15	High output limit, in %
L16	Low output limit, in %
L17	Not Used
L18	Sample interval (T_S) in milliseconds

Figure 9.1 PID Register Allocation

PROGRAMMING CONSIDERATIONS

- The tuning parameter values of **SP**, K_p, K_I, K_D, and **B** are entered with programming equipment or in the user program.

- Registers L5, L6, L7, L14, L15, and L16 are entered to reflect a percent of scale from 0000 to 9999. The decimal point is implied as XX.XX. For example, an OUTPUT of 50% is entered as 5000, a PROCESS VARIABLE value of 7475 represents 74.75% of full-scale, etc.

- Registers L11, L12, and L13 are entered in the range of 0 to 32,767 with the decimal point implied as XXX.XX. For example, a gain of 1.00 is entered as 100, a reset of 27 repeats per minute is entered as 2700, while a rate of 150 seconds is entered as 15000.

- Register L18's (sample interval) implied decimal point is XX.XXX. For example, a sample interval of 200 milliseconds is entered as 200, while a sample interval of 10 seconds is entered as 10,000.

- The high and low output limits are entered in the same format as the tuning parameters, as a percent of full-scale.

- The loop algorithm calculates **SUM (I)** and **OUTPUT**, and stores PV_{n-1} for the next calculation. The user needs to input PV_n and transfer the calculated **OUTPUT** to a real-world analog output register. These manipulations typically occur through ladder rungs.

- Additional ladder rungs are required to set up a total PID function. See the example in Section 9.7.

- In order to ensure the integrity of the calculations, the Model 650 performs limit checks on the variables loaded by the user. Valid ranges are defined as follows:

$0 \leq X \leq 9999$	for OUTPUT, PV, SP, Bias, and High and Low Output Limits.
$X \geq 0$	for K_p, K_I, and K_D.
$X > 0$	for sample interval T_s

  If any of these range limits are not observed, the PID algorithm is not executed and bit 18 of the destination register (the register to the left of the "=" sign in the PID rung) is set to flag the error. In addition, the processor sets bit 18 of the out-of-range register to facilitate troubleshooting by identifying the source of the illegal value.

- All registers in the user-defined PID register block must have addresses in the range of 1 to 8000.

9.3 Reverse-Acting Loop

The SPEC 46 entry in a LET box calculates a reverse-acting (PIDR) loop in the following sequence:

1. $\textbf{SUM (I)} = \dfrac{K_P * K_I * T_S * (SP - PV_n)}{6 * 10^{10}} + \textbf{SUM(I)}$

2. LET L1, L2 = **SUM (I)**

3. $\textbf{OUTPUT} = \left[\textbf{SUM (I)} + \dfrac{K_P * (SP - PV_n)}{10{,}000} + \dfrac{K_P * K_D * (PV_{n-1} - PV_n)}{T_S * 1000} + \dfrac{B}{100}\right] * 100$

4. LET $PV_{n-1} = PV_n$

5. *CLEAR* bit 10 of register 8 (reverse-acting flag)

6. If low output limit exceeds high output limit, set bit 16 of register 8 to flag this illegal condition

7. Check if output exceeds the high output limit. If YES, and $K_I \neq 0$, a back calculation is performed on the integral sum.

   IF **OUTPUT** > High output limit:

   A. Set bit 14 of register 8

   B. LET L1 = $\dfrac{\text{high output limit} - \textbf{OUTPUT}}{100} + \textbf{SUM(I)}$

   C. LET **OUTPUT** = high output limit

8. If **OUTPUT** < (high output limit + 1), reset bit 14 of register 8 to indicate that high limit is not exceeded.

9. Check if output is below the low output limit. If YES,and $K_I \neq 0$, a back calculation is performed on the integral sum.

   IF **OUTPUT** < Low output limit:

   A. Set bit 15 of register 8

   B. LET L1 = $\dfrac{\text{low output limit} - \textbf{OUTPUT}}{100} + \textbf{SUM(I)}$

   C. LET **OUTPUT** = low output limit

10. If **OUTPUT** ≥ low output limit , reset bit 15 of register 8 to indicate that low limit is not exceeded.

11. The final action in the Reverse-Acting Loop is to store the newly-calculated loop **OUTPUT** value in L5 and in the SPEC 46 LET box register which is just to the left of the " = " sign.

9.4 Direct-Acting Loop

The SPEC 47 LET box calculates a direct-acting (PIDD) loop. The algorithm used is identical to the one used for the reverse-acting loop, except that the sign for the mathematical difference between **SP** and **PV** terms is reversed.

The change in sign results in changes to the first, third, and fifth steps of the sequence in Section 9.3. All other steps are the same as shown in Section 9.3.

As with the PIDR, the newly-calculated loop **OUTPUT** value is stored both in register L5 and in the register to the left of the "=" sign in the SPEC 47 LET box.

1. $\textbf{SUM (I)} = \dfrac{K_P * K_I * T_S * (PV_n - SP)}{6 * 10^{10}} + \textbf{SUM(I)}$

2. SAME AS PIDR

3. $\text{OUTPUT} = \left[\textbf{SUM (I)} + \dfrac{K_P * (PV_n - SP)}{10{,}000} + \dfrac{K_P * K_D * (PV_n - PV_{n-1})}{T_S * 1000} + \dfrac{B}{100}\right] * 100$

4. SAME AS PIDR

5. SET bit 10 of register 8 (direct-acting flag)

6. SAME AS PIDR

7. SAME AS PIDR

8. SAME AS PIDR

9. SAME AS PIDR

10. SAME AS PIDR

11. SAME AS PIDR

9.5 Manual Loop

The SPEC 48 instruction in a LET box causes the PID loop to be manually (PIDM) controlled. The number preceding SPEC 48 in the LET box must point to the same block of 18 registers as did the PIDR or PIDD loop algorithm.

The PIDM loop is executed as follows to facilitate a bumpless transfer to AUTO:

1. LET **SP** = PV_n (sets the setpoint equal to the process variable)
2. LET **B** = **OUTPUT** (sets bias equal to the loop output)
3. LET **SUM(I)** = 0 (zeroes out the integral summation)
4. LET $PV_{n-1} = PV_n$
5. The final action in the Manual-Acting Loop is to transfer the **OUTPUT** value in L5 to the SPEC 48 LET box register which is just to the left of the " = " sign.

NOTE: When adjusting the **OUTPUT** while under manual control, the user must make the adjustments to the L5 output register in the 18-register block and not directly to the register assigned to the external output. See Example 1 in Section 9.7.

9.6 LEAD/LAG Functions

NOTE: *This function is also described in Instruction Bulletin #30598-240-0x.*

The SPEC 49 (LEDLAG) instruction in a LET box performs a LEAD/LAG calculation on the process variable (PV). The number preceding SPEC 49 in the LET box is used as a pointer to a user-defined block of seven registers. This number must be an integer from 1 to 7993.

In the following table (Figure 9.2), a "Z" prefix is used to denote the seven storage registers that are used in the LEAD/LAG calculations. Characters in boldface represent variables used in the equations that follow.

The LEAD/LAG algorithm calculates Q_n based on user-entered values for T_s, T_d, and T_i. Since these values are stored in integer registers, they contain an implied decimal point that must be entered in the following format:

For T_d and T_i ..... XXX.XX
For T_s ..... XX.XXX

REGISTER #	FUNCTION
Z1	The resulting calculation for this lead/lag update (Q_n)
Z2	The process variable used in this update (PV_n)
Z3	The sample interval in milliseconds (T_s)
Z4	The lead time constant in hundredths of a second (T_d)
Z5	The lag time constant in hundredths of a second (T_i)
Z6	The resulting calculation from the previous update (Q_{n-1})
Z7	The process variable from the previous update (PV_{n-1})

Figure 9.2 LEAD/LAG Register Allocation

OTHER CONSIDERATIONS

- The process variable (**PV**) must be loaded into register Z2.
- The calculated value of Q_n must be transferred into the PIDD or PIDR loop register L6 (the process variable register).
- The LEAD/LAG algorithm retains Q_{n-1} and PV_{n-1} for the next calculation.
- In order to ensure the integrity of the calculations, the Model 650 performs limit checks on the variables loaded by the user. Valid ranges are defined as follows:

$0 \le X \le 9999$	for PV.
$X \ge 0$	for T_d and T_i.
$X > 0$	for sample interval T_s.

If any of these range limits are not observed, the LEAD/LAG algorithm is not executed and bit 18 of the destination register (the register to the left of the "=" sign in the LEDLAG rung) is set to flag the error. In addition, the processor sets bit 18 of the out-of-range register to facilitate troubleshooting by identifying the source of the illegal value.

See Example 2 in Section 9.7 for typical ladder rungs required.

The SPEC 49 entry in a LET box causes the following steps to occur:

1. $$Q_n = \left[\frac{\frac{T_i}{100} * \frac{Q_{n-1}}{100} + \left[\left(\frac{T_s}{1000} + \frac{T_d}{100} \right) * \frac{PV_n}{100} \right] - \left(\frac{T_d}{100} * \frac{PV_{n-1}}{100} \right)}{\frac{T_s}{1000} + \frac{T_i}{100}} \right] * 100$$

2. LET $PV_{n-1} = PV_n$ (prepare for next calculation)

3. LET $Q_{n-1} = Q_n$ (prepare for next calculation)

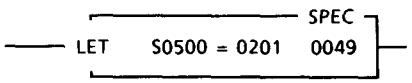
NOTE: *Since the LEAD/LAG algorithm performs a calculation, it is governed by the same rules as any calculation. If the result of the calculation cannot be represented as a 16-bit integer (-32767 to 32767), then numerical overflow errors (refer to Section 7.11, Overflow Errors) will occur. When this happens, the LEAD/LAG calculation is not executed.*

Overflow may occur the very first time the LEAD/LAG function is calculated whenever the lag constant (Z5) is set to '0000'. As a strictly lead calculation, the algorithm simplifies as follows:

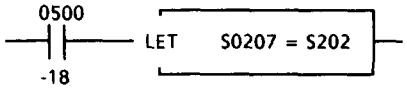
$$Q_n = PV_n + \left[\frac{10 * T_d}{T_s} * \left(PV_n - PV_{n-1} \right) \right]$$

From this relationship it can be shown that if (T_d/T_s) is greater than or equal to '1', then for PV_n - PV_n-1 values greater than 2978 (approximately 29.78 per cent of scale), numerical overflow will occur. The first time the calculation is performed, PV_n-1 is equal to '0000'. It is also possible for signals with a low signal to noise ratio for overflow to occur periodically during normal ladder scanning.

The LEAD/LAG algorithm in lead only mode is a differentiator. A differentiator is designed to amplify signal, with noise included. The overflow condition can be detected and appropriate application action can be taken. Assume the following LEAD/LAG rung:



If overflow occurs in this ladder rung, overflow bit 0500-18 will become energized, and the LEAD/LAG algorithm will not execute. This overflow bit can be used by an application rung to compensate for the overflow condition.



9.7 Application Considerations and Examples

While the loop algorithms in this section are the same as shown in Instruction Bulletins 30598-301-XX and 30598-240-XX, certain differences exist in the way the Model 650 executes and implements the results. For example, although the loop update time is user-specified, the Model 650 scan time and I/O update time must be fast enough to accommodate the loop time.

The concept of *throughput* must be understood, and the Special Instruction Set for the Model 650 (Section 8 of this manual) must be read and understood as well. A working knowledge of PID operation and a good grasp of the terminology contained in two Instruction Bulletins - *30598-301-0X* and *30598-240-0X* - is recommended before proceeding with this section.

Instruction Bulletin #30598-301-0x, entitled Class 8010 PID Closed-Loop Control, Using the SY/MAX Model 300 Programmable Controller, describes the use of a PID control program which can be loaded into a Model 300 or 500 Processor via cartridge tape. The discussions of the PID control equation in Sections 2.0 and 3.1 of that bulletin should be compared to the algorithms for PIDD and PIDR in this bulletin; they are identical except for data on sample interval.

Also, note the similarities between the register usage table in Section 3.2 of Bulletin 30598-301-0X and the usage in the Model 650; loop number, alarm status, and alarm acknowledge registers are not used in the Model 650, but the loop status *does* use several bits. The PV_{n-1}, integral sum Most Significant Digit and integral sum Least Significant Digit registers also differ from the 30598-301-0X bulletin.

Instruction Bulletin #30598-240-0X, titled Class 8040 Type PCM-110 Process Control Module, describes the PID algorithm that is available in the firmware of the PCM. The PID control equation in Figure 8.2 of that bulletin should be compared to the PIDD and PIDR algorithms in this Model 650 bulletin. Note that the PID algorithm A in the PCM bulletin is the same one used for the Model 650.

The register usage table in the PCM bulletin should be compared to the usage of the Model 650. Register differences exist because the PCM, being a dedicated closed-loop controller, has many features not found in the Model 650. References to alarm status, remote setpoint, deadband, etc., as well as listings beyond loop register 15-do not apply to the algorithm used by the Model 650 (although applying a LEAD/LAG function to the process variable does exist in the form of a SPEC 49 LET box). The following examples show how some of these special functions are utilized.

> **CAUTION**
>
> **These examples are intended to be used only as guidelines when implementing the algorithms. It is the user's responsibility to understand these examples before incorporating these or any other rungs into a working program.**

EXAMPLE ONE: Reverse-acting single PID loop (PIDR loop)

GIVENS:

- The setpoint (**SP**) and tuning (K_P, K_I, K_D, and **B**) parameters are already set.
- The process variable *input* is in register 513.
- The process variable *output* is in register 517.
- The 18-register PID definition register block consists of registers 51-68.
- Reverse-acting PID control will be applied based on a 200 ms loop update time.
- The loop calculations must be performed at 200 millisecond intervals. For this example, the necessary repeatability will be achieved with a *timed interrupt* subroutine. Note use of *timed interrupt* precludes RUN-mode programming. If RUN-mode programming must be preserved, some other technique (such as using register 8166 to regulate repeatability) must be used to ensure execution based on the 200 millisecond sample interval time.
- If the loop is not in AUTO, loop control will default to MANUAL.

RUNGS NEEDED:

1. TIMED INTERRUPT RUNG - The following rung must be the *very first rung* in the program if a timed interrupt subroutine is used. It allows the main program to call the subroutine containing the PID algorithm at specified intervals. In this example, the subroutine is called every 200 milliseconds with a tolerance of 15 milliseconds. See Section 6.8.1. Note that use of the timed interrupt subroutine does not allow for RUN-mode programming.

2. SAMPLE INTERVAL RUNG - The following rung establishes the sample interval in milliseconds in register 68 (L18). The interval in this example is 200 ms; this value must agree with the value in rung 1, and represents the frequency with which the routine must be executed.

3. HIGH OUTPUT LIMIT RUNG - The following rung programs the high output limit into register 65 (L15). In this example, the maximum high limit of 9999 is programmed.

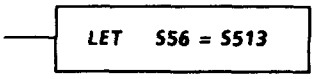
4. LOW OUTPUT LIMIT RUNG - The following rung programs the low output limit into register 66 (L16). In this example, the minimum low limit of 0000 is programmed.

THE MAINLINE USER PROGRAM EXISTS HERE, BETWEEN RUNG "4" ABOVE AND RUNG "A" BELOW

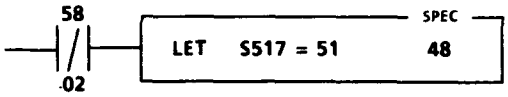
A. SUBROUTINE AREA RUNG - The following rung appears only once in any program, and serves to mark the memory area which holds all subroutines.

B. START SUBROUTINE RUNG - The following rung marks the point at which the timed interrupt subroutine begins.

C. COPY PV RUNG - The following rung copies the process variable from the external input register (513 in this example) into the loop process variable register (56 in this example, defined as L6 in the register block).



D. EXECUTE PIDM CALCULATION RUNG - The following rung begins PIDM calculations if bit 2 in register 58 has been reset; if bit 2 = 0, then loop is in MANUAL. Register 51 is the pointer that both points to the register block 51-68 and also defines register 51 as L1, 52 as L2, etc. Register 517 is this example's external output register, where the calculated output is placed.



E. EXECUTE PIDR CALCULATION RUNG - The following rung begins PIDR calculations if bits 2 and 3 in register 58 have been set. These two bits correspond to the "mode status" bits previously described. Bit 2 set to "1" indicates the loop is NOT in manual, while bit 3 set to "1" indicates the loop is in the AUTO mode. Again, register 51 is the pointer that both points to the register block 51-68 and also defines register 51 as L1, 52 as L2, etc. Register 517 is this example's external output register where the calculated output is placed. To change this loop to a PIDD (direct) loop, only change SPEC 46 to SPEC 47 in the following rung.

F. RETURN FROM SUBROUTINE RUNG - The following rung identifies the end of the timed interrupt subroutine.

EXAMPLE ONE PROGRAMMING CONSIDERATIONS

- The execution time of the timed interrupt subroutine should be kept as short as possible. The rungs that establish the loop initialization constants (for *sample interval, high output limit* and *low output limit*- rungs 2, 3 and 4 in example 1) are programmed in the mainline area of the ladder program.

- Rung C in the subroutine (which transfers the process variable into register L6) is executed immediately prior to the PID rungs (D and E) so that the most recent input update is used in the PID computations.

- Bit 2 of register 58 controls whether the loop is under manual or automatic control. In the Example One program, the loop can be disabled (no PID computations performed) if bit 2 of register 58=1 (loop not in MANUAL) and bit 3 of register 58=0 (loop not in AUTOMATIC).

- The order in which rungs D and E are programmed is irrelevant.

- The **OUTPUT** for rungs D and E also exists in register 55 (L5). Programming the external output register 517 avoids programming the additional rung: *LET S517 = S55*.

- If the loop is being executed in MANUAL (bit 2 of register 58=0), the output must be adjusted in register 55, *NOT* in register 517. For example, to set the output to 50% of full-scale range, the value 5000 is entered into register 55. When the Model 650 executes rung D, the contents of register 55 are transferred into register 517. If 5000 had been entered into register 517, it would have been overwritten by the current contents of S55 in the transfer operation.

- The worst-case execution time for the block of rungs within the timed interrupt subroutine is approximately one millisecond.

- There is no absolute limit to the number of PID loops that the Model 650 can execute. Practical limitations for a given system are a function of the number of available registers, system throughput, and the user's ability to manage multiple loops. Additional PID loops can be executed by either adding the necessary blocks of rungs into the MARK ST. SUB area or by multiplexing the loop algorithm by changing the pointer from a constant to a variable and the output register from a single address to a matrix. The first method is simple; the second requires experience in advanced programming.

- System throughput is decreased for every PID loop that is added to the program. The interrupt rate must be infrequent enough to allow the Model 650 to scan the remainder of the program in a reasonable amount of time for the application.

- If the PID algorithm fails to execute as expected, check to see if bit 18 of register 517 is set to indicate an out-of-range condition. If set, monitor the other registers in the block to determine which register is out of range, as indicated by bit 18 of the offending register.

- If the execution time of the timed interrupt subroutine exceeds the interrupt rate, ERROR 29100 is generated and the Model 650 halts.

- If the interrupt tolerance is too restrictive, ERROR 29102 is generated and the processor halts.

EXAMPLE TWO: LEAD/LAG Function Used With Example One Program

NOTE: *Instruction Bulletin 30598-240-XX contains more information on the LEAD/LAG function.*

Assume the same conditions exist as in Example One except that the LEAD/LAG function is to be applied. Additional given conditions for the LEAD/LAG portion of the program are therefore required.

LEAD/LAG GIVENS:

- The process variable exists in register 513.
- Registers 71-77 (Z1-Z7) make up the block that defines the LEAD/LAG function.
- The sample interval T_s is preset in register 73 (Z3).

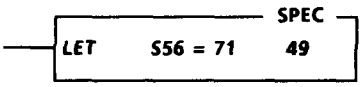
NOTE: *The user must enter the LEAD/LAG time constants (T_D and T_I) in registers 74 (Z4) and 75 (Z5) respectively.*

The following rungs must appear in the timed interrupt subroutine and replace "RUNG C" as shown in the previous Example One in the first example:

The following rung presets the sample interval (in milliseconds) into register 73 (Z3). This number must correspond to the timed interrupt interval specified in rung 1 and also match the interval specified for the PIDR or PIDD algorithm.

The following rung loads the process variable into register 72 (Z2) for use in the LEAD/LAG calculation.

The following rung performs the LEAD/LAG function on the process variable currently loaded in register 72 (Z2). "71" is the pointer that points at registers 71-77. The calculated Q_n is placed in register 56 for use in the upcoming PIDM or PIDR calculations.



10 ASCII COMMUNICATIONS (Input and Output)

10.1 ASCII Input

NOTE: *Since ASCII Input may not be assigned to the Ethernet port (Channel 3), this section applies only to ports 1 and 2 (PRGMR and COMM ports).*

The Model 650 uses an enhanced PRINT rung format to allow inputting of ASCII-coded data. Termination of the ASCII input string can be done either by character *count*, character *delimiter*, or both.

10.1.1 DESCRIPTION/INITIATION

The ASCII input programming format is similar to that of ASCII output. A special register address (S8006) is inserted as the first register in the PRINT statement and is used to identify the mode. Subsequent entries in the PRINT statement further define the function. The ASCII input rung must be programmed *with all entries* as shown in Figure 10.1:

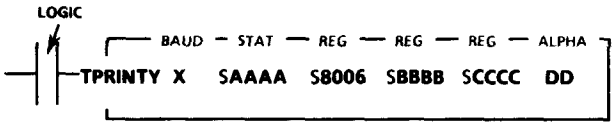


Figure 10.1 ASCII Input Rung

Each rung parameter in Figure 10.1 is explained as follows:

LOGIC upon becoming TRUE, the port specified by Y is enabled to accept ASCII input characters. When FALSE, the port ignores the ASCII input and reverts to SY/MAX protocol communication, unless a complete ASCII input string had not been received when previously TRUE. Refer to Section 10.1.3.

Y is the CHANNEL that receives the input, being either 1 (PRGMR port) or 2 (COMM port). The serial word structure for channel 1 is defined in register 8099. The structure for channel 2 is in register 8100. See Figure 10.3.

X is the BAUD RATE for the selected channel (Y). See Figure 10.5 for available baud rates.

SAAAA is the STATUS REGISTER assigned to this communication rung. DO NOT use this user-specified register for any other purpose in the program. Error codes that may appear in the status register are listed in Section 10.1.5.

S8006 is not actually a register, but is the COMMAND that defines the box as an ASCII input box.

SBBBB identifies the STARTING REGISTER for storing the data block. The block can hold no more than 256 ASCII characters. If more than 256 characters are input, an error is sent to the status register and ERROR 19 is posted. *This register is the start address, not a pointer to a start address.* Each register in the block will contain a single character (stored in the least significant byte).

SCCCC identifies the register whose contents specify the number of characters to be inputted. When this number of ASCII characters has been received, the ASCII input string is terminated and bit 16 of the associated status register is set ON.

DD are the contents of the ALPHA box that identifies the ASCII character which the user specifies as the *block terminator* (delimiter). The desired terminator character should be loaded, followed by a space (space bar) character to shift the character into position. When this character has been received, the ASCII input string is terminated and bit 16 of the associated status register is set ON.

CAUTION

At the end of every block of received data, the Model 650 appends a null (00) character to mark the end of the block. Due to this feature, the user MUST reserve one additional register beyond the maximum character count for the block size. If insufficient registers are reserved, incoming data may overwrite register data that is used for other purposes in the program.

NOTES:

1. *If no characters are entered for* **DD** *in the ALPHA box, the default block terminator is a space (20H).*

2. *Setting the contents of* **SCCCC** *to zero defaults to the maximum input block size of 256 characters.*

3. *If both* **SCCCC** *and* **DD** *are specified, the ASCII input block will stop upon reaching the first condition to be satisfied.*

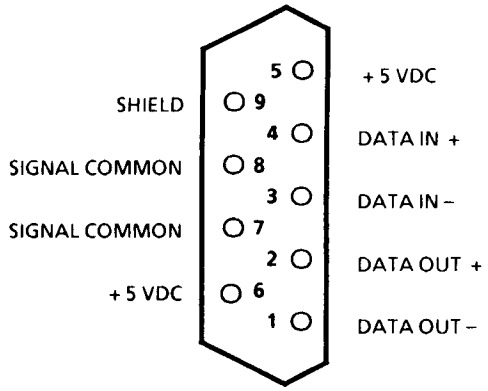


Figure 10.2 PRGMR and COMM Port Pinout

10.1.2 PORT CONFIGURATION

Both the PRGMR and COMM ports of the Model 650 are configured as RS-422 ports. The pinout for these ports is shown in Figure 10.2.

NOTE: *RTS and CTS hardware handshaking signals are not supported. Pins 5, 6, 7, and 8 are used for + 5VDC and signal commons.*

When configuring each of the PRGMR and COMM ports for ASCII input, registers 8099 (for PRGMR) and 8100 (for COMM) are used. Figure 10.3 illustrates what each of the bits in these two registers is used for and how to configure the port for the desired word structure.

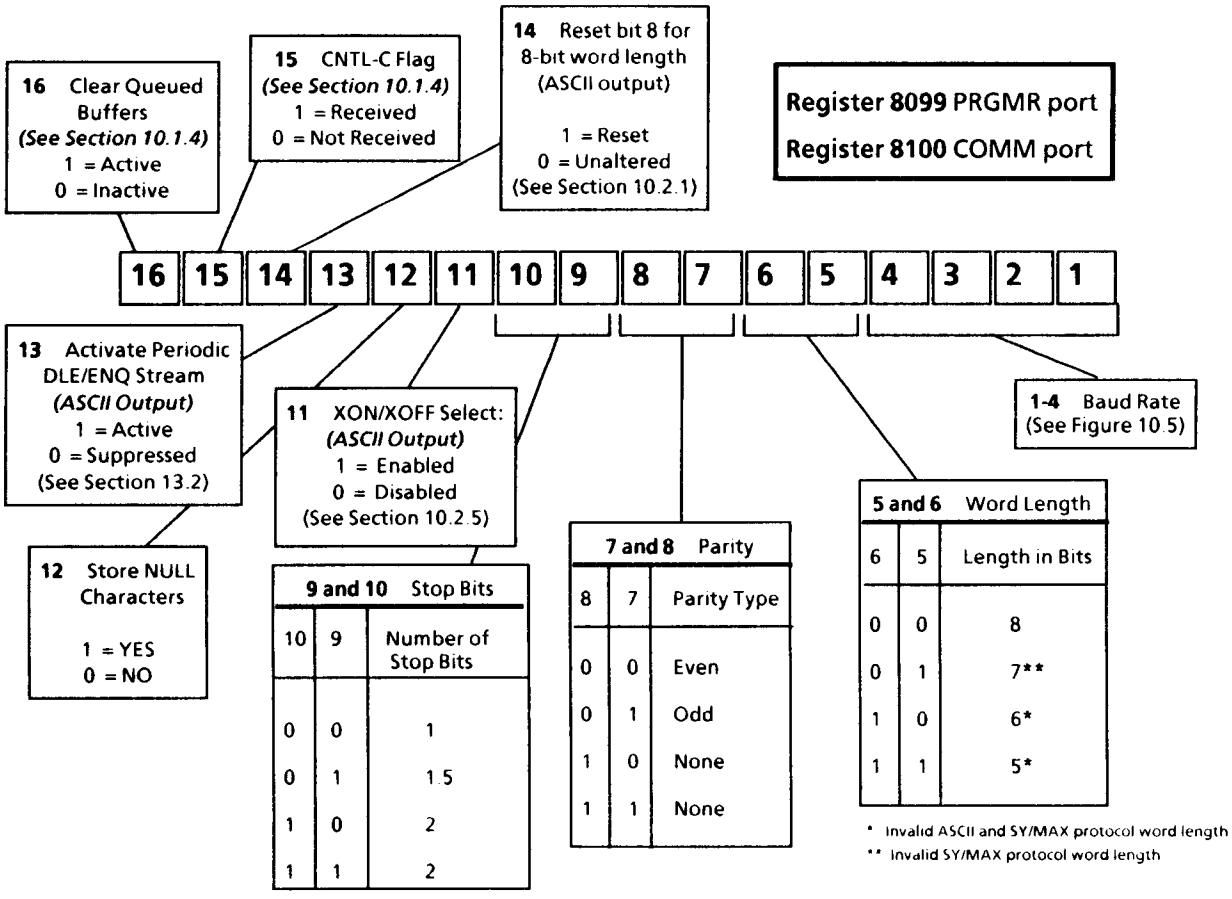


Figure 10.3 PGRMR and COMM Port Configuration Registers

10.1.3 OPERATION

When the logic preceding the ASCII input box transitions from FALSE to TRUE, the specified port is enabled to receive ASCII, queueing up an input buffer. Once enabled, the port remains enabled to accept an ASCII input string even if the logic preceding the ASCII input box becomes FALSE.

Incoming characters are then stored in a communication port buffer until a complete ASCII string has been received. As soon as the final character is received, the port buffer ignores future ASCII characters. These characters are lost unless another port buffer has been queued to accept them. A total of 11 buffers exist for both ports. Of these, up to 9 can be allocated by the processor to a single port. This makes it possible to anticipate multiple ASCII input strings by enabling multiple ASCII input rungs, or even by repeatedly transitioning the logic of a single rung, since a new buffer is allocated (queued) upon every transition.

The port buffer is then read by the processor during its end-of-scan communication port servicing and the ASCII characters are moved into the registers specified in the ASCII input rung.

			RESULTING BIT STATUS		
LOGIC	BUFFER QUEUED	PORT ASCII INPUT STATUS	BIT 16	BIT 17	BIT 22
FALSE	NO	INPUT DISABLED	0	0	0
TRUE	YES	INPUT ENABLED	0	1	1
TRUE	NO	INPUT COMPLETE	1	1	1
FALSE	YES	INPUT ENABLED	0	0*	0
TRUE	NO (all previously allocated)	INPUT ENABLED	X (don't care)	0	1

** When LOGIC is FALSE, status bits are cleared but buffer remains in queue.*

Figure 10.4 Bit Status vs Logic and Buffer States

BAUD RATE CODE		
BINARY BITS 4321	DECIMAL	BAUD RATE
0000	0	50
0001	1	110
0010	2	300
0011	3	1200
0100	4	2400
0101	5	4800
0110	6	9600
0111	7	19.2K

Figure 10.5 Baud Rate vs Bit Pattern (See Figure 10.3)

Bits in the status register reflect the above situations; bit 22 is set to "1" when the input logic is TRUE, bit 17 is set when a port buffer is enabled for input, and bit 16 is set when a complete ASCII input string has been received and processed by the CPU. Figure 10.4 shows which status bits are set based on the ASCII input rung's current logical state and whether a port buffer is queued for ASCII input from a *previous* FALSE-to-TRUE logic transition.

Buffers become "linked" with the ASCII input rungs in the order in which they are executed TRUE, *unless all 9 buffers have been allocated.* In this case, even though status register bit 22 is set, bit 17 *is not.* This indicates that all buffers are queued. A buffer is not freed up until a complete ASCII string has been received and the buffer has been read by the Model 650 during its end-of-scan activities.

NOTE: *If all 9 buffers are simultaneously used for ASCII input while priority SY/MAX communications (READ, WRITE, ALARM) are transmitting out the other port, port 2 (COMM) must be used for ASCII input; port 1 (PRGMR) is used for the SY/MAX communication.*

10.1.4 APPLICATION CONSIDERATIONS

- ASCII storage registers contain the binary code for the ASCII character, not the character itself. For example, the ASCII character "0" is represented as "48" (decimal), where an ASCII "A" is represented as "65" (decimal).

- One ASCII character is stored per register (in the least significant byte).

- ASCII input information does not appear in the storage registers until a complete block, defined by the block size or the terminator, has been received.

- Null (00) characters may either be disregarded or stored as valid characters. See bit 12 in Figure 10.3 .

- If null characters are disregarded, the end of the input register block is defined either by the block terminator or by the first register to contain a null character.

- The following description applies to bits 15 and 16 of port attribute registers 8099 and 8100.

  Bit 16 - when set, aborts any pending ASCII input instructions and clears any port buffers queued for an ASCII input string. This is a means of error recovery when characters in an ASCII input string are lost or garbled. Setting bit 16 to "1" via a coil in the ladder program gives the user a means of purging the input buffers in preparation for a new ASCII string.

  NOTE: ***In order to work properly, bit 16 MUST be set via a coil in the ladder program and NOT by any other means, such as a LET or via DATA ENTER, COMMs, etc.***

  Bit 15 - acts as a flag that recognizes an ASCII input BREAK command. Upon receiving a CTRL-C (03H), bit 15 latches ON. For example, when programmed as a contact to control bit 16, a CTRL-C character issued from the ASCII output device purges any queued ASCII input buffers.

- Both bits 15 and 16 of the port attribute registers are reset to "0" upon power-up.

- Upon a power cycle, port 1 (PRGMR) resets to 9600 baud, 8-bit word, even parity, with one stop bit in order to be compatible with the SY/MAX protocol. Port 2 (COMM) retains any changes made to its attributes.

- DEL/CLR ALL does *not* cause port attributes to reset to the default values *unless* prompted by a "CLR MEMORY NEEDED".

- Because valid ASCII characters are defined as seven bits long, the Model 650 (when configured to receive an *eight*-bit ASCII word) masks off the state of bit 8 in the incoming character.

- Halting the Model 650 or setting bit 16 of register 8099 or 8100 to "1", while awaiting an ASCII input string, clears any queued port buffers, halts the ASCII input instruction, and generates ERROR 17 in the status register.

- The ASCII input function only operates properly when the device that is outputting the ASCII characters is *directly* connected to a Model 650 port. ASCII sent over the network is not accepted by the ASCII input function.

- When a port is queued for ASCII input while FORCING is active, the "ASCII port" is assumed to be dedicated to the ASCII input function. If the cable is removed from the *other* communication port, no DLE/ENQ characters are transmitted out the ASCII port to inquire if a programming device is present.

- If a port's word structure is altered, it *remains* altered even after the port reverts to SY/MAX protocol. SY/MAX protocol requires an 8-bit word with even parity, and one stop bit, to operate correctly.

- The ASCII input data block (of up to 256 characters) must be stored in register addresses 8000 or less.

- The status register used in the ASCII input rung MUST have a register address of 8000 or less.

10.1.5 ERROR CODES

The error codes shown in Figure 10.6 below apply to the ASCII input operation and may appear in the status register of the ASCII input rung ("SAAAA", in Figure 10.1).

ERROR NUMBER	DESCRIPTION
11	Receiver overflow. The Model 650 is unable to handle incoming characters fast enough, due to the servicing of other interrupts.
13	Parity error
15	Framing error. Typically due to baud rate or word structure problems.
17	Input was terminated before a complete ASCII string was received. Results from either the CPU going to HALT or bit 16 of register 8099 or 8100 being set. If bit 16 is set, buffers queued for ASCII input are cleared.
19	Buffer overflow. The 256-character buffer was filled before the terminating character was received.

Figure 10.6 ASCII Input Error Codes

10.2 ASCII Output

The capability for outputting ASCII characters via PRINT statements exists for all Model 650 ports. Refer to the appropriate programmer instruction bulletin 30598-167, 30598-174 or 30598-717 for general information on programming PRINT rungs.

10.2.1 STANDARD ASCII FORMAT

The standard ASCII characters are output as RS-422 signals (see Figure 10.2 for port pin-out) defaulting to 8-bit words with even parity and 1 stop bit, transmitted at 9600 baud.

NOTE: *Since ASCII output may be assigned only to ports 1 and 2 (PRGMR and COMM ports), the baud rate and ASCII output considerations do not affect port 3 (Ethernet port).*

If the receiving device does not automatically mask off bit 8 (typically indicated when half the characters are received correctly, the other half garbled), set bit 14 of register 8100 for port 2 (register 8099 for port 1) to 1 to cause the processor to perform the mask. Another alternative to match output default settings is to set up the receiving device for a 7-bit word with odd parity and 2 stop bits. Other settings may be accommodated by adjusting registers 8099 or 8100; refer to Figure 10.3.

Standard PRINT statements allow outputting fixed ASCII characters (represented by ALPHA entries in the PRINT rung) and register contents (represented by REG entries in the PRINT rung). This allows an English language message to be augmented with variable register data. Each REG entry will result in 6 ASCII characters being transmitted, representing up to 5 decimal places plus sign (if negative) or space (if positive). Leading zeros will automatically be transmitted as spaces. Each standard ASCII PRINT statement automatically appends a CR and LF to the ASCII string, unless suppressed with a semicolon (;) in the last ALPHA position.

10.2.2 RAW BINARY ASCII FORMAT

While the standard ASCII format will handle the majority of ASCII output applications, some of the special CONTROL CODES (ESC, DLE, etc.) cannot be accommodated by the ALPHA entries. Any 7-bit ASCII code (0 through 127) can be output via the Raw Binary ASCII format.

To invoke the Raw Binary format, utilize the same PRINT statement as for the standard format with 2 exceptions: REG S8004 must be the first element following STAT in the horizontal PRINT box (refer to Figure 10.7 below) and all subsequent elements of the box should be REG, not ALPHA.

To output the ASCII codes, store the desired 8-bit patterns in the registers specified by the REG elements. The data in each register will be treated as a pair of 8-bit characters (low-byte bits 1-8, high-byte bits 9-16) with each byte ranging from 0 through 127. The high-byte character is transmitted first, followed by the low-byte character. If the contents of the register is a single byte (0-127), then that byte, representing a single ASCII character, is transmitted.

EXAMPLE: This example demonstrates the use of register 8004 in a print rung to set the Model 650's "raw binary" output format.

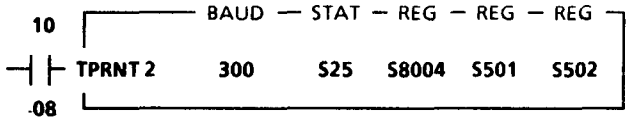


Figure 10.7 Raw Binary Example

To output a carriage return (CR), a line feed (LF), and the number "1", three bytes are required. This requires assigning two registers - the first register contains the CR in the high byte and the LF in is the low byte. The second register's low byte contains the number "1". Registers 501 and 502 are used in this example to contain the ASCII codes which can be found in Figure 10.8.

A suggestion on how to store these values is to multiply the desired high byte of register 501 (the CR is ASCII 13) by 256 then add the low byte (the LF, ASCII 10) to the product. This yields 3338 which is stored in register 501. This leaves the single byte (the number "1"); store its ASCII value (49) directly in the low byte of register 502.

When the contact 10-08 is closed in the above rung, a CR followed by a LF followed by the number "1" will be sent out of port 2 at 300 baud.

APPLICATION CONSIDERATIONS

- Default format is standard ASCII for each new PRINT rung.
- Multiple formats may be used in the same rung by entering a REG value of S8001 (for standard ASCII) or S8004 (for Raw Binary ASCII).
- Raw binary format character entries should be REG elements, not ALPHA.
- The processor automatically appends a CR and LF at the end of any standard ASCII PRINT statement unless suppressed by a semicolon; no characters are appended to a Raw Binary ASCII PRINT statement.
- Data is transmitted as a 7-bit ASCII word with odd parity and 2 stop bits, except for Raw Binary format, which is an 8-bit data word with even parity and 1 stop bit.
- BAUD rates up to 19.2 K are allowed, and BAUD RATE should be specified in each PRINT rung.
- Each PRINT rung must have a unique status register.
- If a PRINT rung is programmed to transmit no data, a single NULL (00 hex) will be transmitted.

10.2.3 MISCELLANEOUS ASCII FORMATS

Up to this point, the REG elements in a PRINT box could be output as a standard decimal value (5 digits plus sign in the standard ASCII format) or as an ASCII character representation (in the "raw binary" ASCII format). It is also possible to print out the register contents in 2 other formats: a 16-digit binary representation of the register bit pattern (all characters either "0" or "1") or a 4-digit hexadecimal representation (characters 0 through F).

To invoke the 16-digit binary representation, have REG S8002 be the first element following STAT in the horizontal PRINT box; for the 4-digit hexadecimal representation, use REG S8003 as the first element. All subsequent REG elements of the PRINT box will be output in the respective format.

NOTE: *To cause a PRINT box format to revert to the standard ASCII default of 5 digits plus sign, replace S8002 or S8003 with S8001.*

10.2.4 REPEATED CHARACTER STRINGS

For applications which require outputting the same character numerous times, another special format is available. By having REG S8005 as the first element following STAT in the horizontal PRINT box, subsequent REG elements will be interpreted such that the low byte represents the ASCII character (0-127) and the high byte the number of times this character is to be repeated (up to a maximum of 192). If the high byte = 0 or is greater than 192, no data is sent.

10.2.5 XON/XOFF HANDSHAKING

For all PRINT formats, the Model 650 can be enabled to support XON/XOFF software handshaking while outputting ASCII. Setting bit 11 of register 8099 (PRGMR port) or 8100 (COMM port) to 1 allows a receiving device (a printer, for example) attached to the respective port to instruct the Model 650 to stop sending ASCII. The device does this by sending a "DC3" (13H or CTRL-S) to the Model 650. When the receiving device is again ready to accept ASCII, it sends a "DC1" (11H or CTRL-Q) to the Model 650.

NOTE: *When XON/XOFF is active while SY/MAX communications (READ, WRITE, ALARM) are being transmitted out the other port, port 2 (COMM) must be used for the ASCII function; port 1 (PRGMR) or port 3 (Ethernet) must therefore be used for the SY/MAX protocol device.*

10.3 ASCII Hex/Decimal Conversion

Figure 10.8 provides a conversion table for ASCII characters in hexadecimal and decimal numbers.

ASCII Character	VALUE Decimal (Hex)	ASCII Character	VALUE Decimal (Hex)	ASCII Character	VALUE Decimal (Hex)	ASCII Character	VALUE Decimal (Hex)
NUL	00 (0000H)	Space	32 (0020H)	@	64 (0040H)	'	96 (0060H)
SOH	01 (0001H)	!	33 (0021H)	A	65 (0041H)	a	97 (0061H)
STX	02 (0002H)	"	34 (0022H)	B	66 (0042H)	b	98 (0062H)
ETX	03 (0003H)	#	35 (0023H)	C	67 (0043H)	c	99 (0063H)
EOT	04 (0004H)	$	36 (0024H)	D	68 (0044H)	d	100 (0064H)
ENQ	05 (0005H)	%	37 (0025H)	E	69 (0045H)	e	101 (0065H)
ACK	06 (0006H)	&	38 (0026H)	F	70 (0046H)	f	102 (0066H)
BEL	07 (0007H)	'	39 (0027H)	G	71 (0047H)	g	103 (0067H)
BS	08 (0008H)	(	40 (0028H)	H	72 (0048H)	h	104 (0068H)
HT	09 (0009H)	)	41 (0029H)	I	73 (0049H)	i	105 (0069H)
LF	10 (000AH)	*	42 (002AH)	J	74 (004AH)	j	106 (006AH)
VT	11 (000BH)	+	43 (002BH)	K	75 (004BH)	k	107 (006BH)
FF	12 (000CH)	,	44 (002CH)	L	76 (004CH)	l	108 (006CH)
CR	13 (000DH)	-	45 (002DH)	M	77 (004DH)	m	109 (006DH)
SO	14 (000EH)	.	46 (002EH)	N	78 (004EH)	n	110 (006EH)
SI	15 (000FH)	/	47 (002FH)	O	79 (004FH)	o	111 (006FH)
DLE	16 (0010H)	0	48 (0030H)	P	80 (0050H)	p	112 (0070H)
DC1	17 (0011H)	1	49 (0031H)	Q	81 (0051H)	q	113 (0071H)
DC2	18 (0012H)	2	50 (0032H)	R	82 (0052H)	r	114 (0072H)
DC3	19 (0013H)	3	51 (0033H)	S	83 (0053H)	s	115 (0073H)
DC4	20 (0014H)	4	52 (0034H)	T	84 (0054H)	t	116 (0074H)
NAK	21 (0015H)	5	53 (0035H)	U	85 (0055H)	u	117 (0075H)
SYN	22 (0016H)	6	54 (0036H)	V	86 (0056H)	v	118 (0076H)
ETB	23 (0017H)	7	55 (0037H)	W	87 (0057H)	w	119 (0077H)
CAN	24 (0018H)	8	56 (0038H)	X	88 (0058H)	x	120 (0078H)
EM	25 (0019H)	9	57 (0039H)	Y	89 (0059H)	y	121 (0079H)
SUB	26 (001AH)	:	58 (003AH)	Z	90 (005AH)	z	122 (007AH)
ESC	27 (001BH)	;	59 (003BH)	[	91 (005BH)	{	123 (007BH)
FS	28 (001CH)	<	60 (003CH)	\	92 (005CH)	|	124 (007CH)
GS	29 (001DH)	=	61 (003DH)	]	93 (005DH)	}	125 (007DH)
RS	30 (001EH)	>	62 (003EH)	^	94 (005EH)	~	126 (007EH)
US	31 (001FH)	?	63 (003FH)	_	95 (005FH)	DEL	127 (007FH)

Figure 10.8 ASCII Hex/Decimal Conversion

11 SEQUENCER

11.1 Description and Initiation

The Model 650 processor is equipped with a Sequencer function that allows the output of information based on time driven and/or event (input) driven conditions.

OPERATIONAL ATTRIBUTES

- Up to 255 steps can be defined in a sequence.
- Up to 64 outputs can be controlled.
- Stepping can be performed forward, reverse, or non-sequentially.
- Automatic or manual operation is allowed.
- An automatic step advance can be based on time, events (up to 16 inputs), or both.
- Time accumulated during a step can be held (paused).
- Output state change can be inhibited despite step completion.
- Individual step times can be programmed with a resolution of 0.1 second.
- A sequence can be repeated, or halted after a single pass.
- As many Sequencers can be programmed as allowed by register availability.

The Sequencer is initiated via the SPEC 65 function command. This command points to a block of registers which contain the operating parameters of the Sequencer.

The minimum rungs needed to program *one* Sequencer are shown in Figure 11.1.

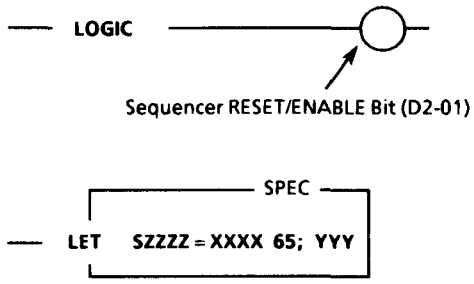


Figure 11.1 Sequencer Initiation Rungs

The rung variables in Figure 11.1 are:

LOGIC Logic in the first rung allows the Sequencer to be reset, or enabled to operate.

SZZZZ A dummy output register. The data in this register will always be the current step number. The recommended programming parameter is: *ZZZZ = XXXX*.

XXXX A value that points to the definition register block. This value must be a constant and range from 1 to 7984. Subtract 6 more for each additional step. i.e., the next step would be 7984-6 = 7978.

SPEC
65 SPECial Instruction #65. Specifies the Sequencer function.

YYY The number of steps in the Sequencer. Must be a constant from 1 to 255.

NOTE: *Placing logic in front of the LET box could result in the Sequencer not being executed.*

11.2 Register Block Allocation

The overall length of each Sequencer register block is variable and changes depending on the number of steps programmed into the Sequencer.

The first ten registers in the block are always the *configuration table registers* and serve to define conditions for *all* steps. Each step also requires six registers to define itself.

Figure 11.2 illustrates how the Sequencer registers are allocated. Since each step used within the Sequencer requires six registers, the total number of registers required per Sequencer is calculated as follows:

[6 × (number of steps) + 10]

See Section 11.2.1 for information on the configuration table registers and 11.2.2 for the step table registers.

NOTE: *The "D" prefix is intended to convey the idea that the register numbers in Figure 11.2 represent relative positions – NOT fixed register numbers.*

11.2.1 CONFIGURATION TABLE REGISTERS

The first ten registers in each Sequencer register block are the configuration table registers. Their function, which affects all steps in the Sequencer, is described in the table shown in Figure 11.3.

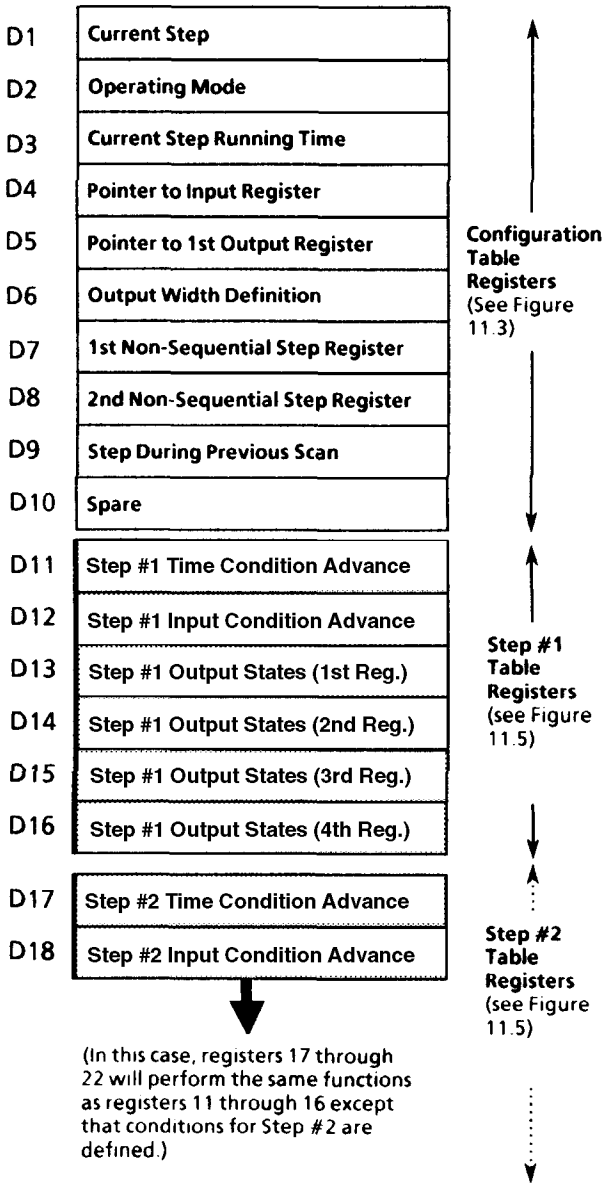


Figure 11.2 Register Allocation

REGISTER #	PURPOSE
D1	*Current step number*. Indicates which step is presently executing. If changed by the user, the sequencer will *immediately* go to the indicated step.
D2	*Operating mode* register. This register defines the sequencer's operating conditions. See Figure 11.4 for a definition of each bit in this register, and Sections 11.4.1 and 11.4.2 for descriptions of the status bits.
D3	*Amount of time* that the current step has been executing. When this register is greater than or equal to the "step time" condition advance (D11, D17, etc), a step advance based on time is enabled
D4	*Pointer to the input register*. Bits in the register being *pointed to* will be compared with the "step input" condition advance (D12, D18, etc.). When the pattern of specified "1"s is matched, a step advance based on input conditions is enabled. If contents are zero, sequencer operates strictly on time condition advance.
D5	*Pointer to the first output register*. The contents of this register point to (define) the first register address of the output field. Bits from the first register of the respective "step output" states (D13, D19, etc.) will be written to the register being pointed at by the contents of D5.
D6	*Bit width of output data*. Valid range is from 1 to 64. This register defines how many bits (beginning with the output register defined by register D5) make up the output field. A value greater than 16 will begin with the low-order bits of the next consecutive register; for example, if register 21 is pointed to as the first output register, and output width = 18, sequencer outputs will be written to register 21 *and* the first 2 (least significant) bits of register 22. Bits 3-16 of register 22, along with registers 23 and 24, are unaffected.
D7	*Non-sequential step register*. Setting bit 8 of the operating mode register D2 causes the sequencer to jump to the step loaded by the user in this register. Note that this jump occurs when conditions for advancing out of the current step are satisfied.
D8	*Same as D7* except that D8 works in conjunction with *bit 9* of the operating mode register D2. Registers D7 and D8 function identically; two registers are provided for flexibility and ease of use, specifically to avoid software timing problems or race conditions.
D9	Used by the Model 650 to keep track of the step executed during the previous scan. If this register is not equal to the current step register (D1), the sequencer *immediately* goes to the new step pointed at by D1.
D10	*Spare (not presently used)*

Figure 11.3 Configuration Table Registers

REGISTER D2 BIT #	PURPOSE
1	**Reset/Enable Sequencer** ("1" = ENABLE, "0" = RESET) When initially enabled, sequencer begins with step #1. Outputs pointed to by registers D5 and D6 will assume the state dictated by Step Register D13 (and D14–D16 if D6 ≥ 17).
2	**Sequence Step Direction** ("1" = reverse, "0" = forward)
3	**RUN/PAUSE Select** ("1" = pause, "0" = run) *When this bit = 0, elapsed time for a* step is allowed to accumulate. When bit = "1", time will *not* accumulate.
4	**Automatic Step Advance Inhibit**. ("1" = hold data, "0" = advance step) When bit = 0, outputs will assume the next state when conditions for an automatic step advance have been satisfied. When bit = 1, outputs will retain their previous state whether or not the automatic step advance conditions have been satisfied.
5	**AUTO/MANUAL Select** ("1" = manual operation, "0" = automatic operation) When in AUTO mode, sequencer is controlled by time/input advance information. When in MANUAL, sequencer advance is controlled by bit 7.
6	**SINGLE PASS/REPEAT Select** ("0" = single-pass, "1" = auto recycle) When bit = 0, sequencer will execute the sequence one time. At the conclusion of the sequence (the last step), all outputs will reset and the "current step" register D1 will indicate "0". When bit = 1, sequencer will proceed to step #1 after meeting all required conditions for advancing beyond the last step.
7	**Manual Advance** ("1" = advance, "0" = maintain) If bit #5 = 1, toggling this bit from "0" to "1" will advance the sequencer.
8	**Sequential/Non-sequential** ("1" = next step not sequential, "0" = next step sequential) When set to "1", the next step advance will be to the step indicated in D7.
9	Same as bit 8, except that the next step is indicated in register D8.
10	**Time and/or Input condition Advance** ("1" = OR, "0" = AND). When "0", step advance is enabled when input condition is satisfied ONLY AFTER the time advance condition has been met. When this bit = "1", the step advance is enabled when *either* the time or input condition has been satisfied.
11–16	Spares (not presently used)

Figure 11.4 Operating Mode Register (D2) Bit Definition

REGISTER D2 BIT #	PURPOSE
17*	Sequence completed.
18*	ERROR bit. See Section 11.4.1.
19*	Manual advance history
20*	Time advance enabled.
21*	Input advance enabled.
22*	Reset/enable history
23*	Step advance just occurred.

* Read-Only Status Bit - See Sections 11.4.1 and 11.4.2

Figure 11.4 Operating Mode Register (D2) Bit Definition (continued)

11.2.2 STEP TABLE REGISTERS

The *step table registers* apply only to a particular step in a sequence. The fixed size for each step table register block is six (see Figure 11.2).

Since the step registers begin immediately after the configuration registers, the first block of step registers occupy the 11th through the 16th register in the Sequencer's overall block of registers. Additional step table register blocks occupy the 17th through the 22nd, the 23rd through the 28th, etc as needed.

Figure 11.5 shows the purpose for each of the six step table registers. Each block of step table registers contains the same parameters – i.e., register 11 defines the same parameter for the first step register block as does register 17 for the 2nd step register block. Similarly, register 12 is the same as register 18, register 13 matches register 19, etc.

REGISTER #	PURPOSE
D11/D17/...	**Time Condition Advance Register.** *Contains the user-specified step time of* this step. Entered in 0.1 second increments. When this time is less than or equal to the Current Step Running Time Register D3, a step advance based on time is enabled. If this register = 0, then the step is strictly input-driven.
D12/D18/...	**Input Condition Advance Register.** Indicates the bits that need to be set before a step advance based on an input condition is enabled. If this register = 0, then step advance is based solely on time. If a step advance is contingent on both time and inputs, normal sequencer operation is such that a comparison for inputs will not be made until *after* the time condition has been satisfied (modifiable by bit 10, register D2).
D13/D19/...	**Defines the output states** of the first output register for each respective step. These bits are written to the "pointed at" register from the *first* output register for the step currently being executed. **NOTE:** *Either register D11/17/... or D12/18/... must contain non-zero numbers before this register is relevant. If both of the conditional advance registers = 0, the step is advanced through.*
D14/D20/...	Same as register D13/D19/..., except that *these are the output states for the second* register (bits 17-32). This register is only relevant if output width (D6) ≥ 17.
D15/D21/...	Same as register D13/D19/... except that these are the output states for the *third* register (bits 33 to 48). This register is only relevant if output width (D6) ≥ 33.
D16/D22/...	Same as register D13/D19/... except that these are the output states for the *fourth* register (bits 49 to 64). This register is only valid if output width (D6) ≥ 49.

Figure 11.5 Step Table Registers

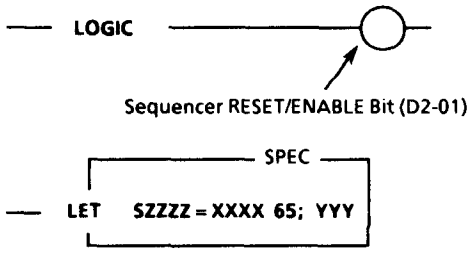
11.3 Application Discussion

The Sequencer provides flexibility for tailoring Sequencer operation to specific needs. This translates into developing a basic sequential algorithm which can then be modified to perform either semi- or non-sequentially. Up to this point the functions of the registers and control bits have been mechanically presented. The purpose of this section is to discuss, from an application standpoint, how to put all of these bits and registers to work in practical situations.

11.3.1 REQUIRED ENTRIES

Regardless of how the Sequencer will be used, the following are *required* actions.

1. The two rungs shown in Figure 11.1 *must* be entered. Since the LET rung must be executed every scan (even when the Sequencer is reset), there must not be any logic in front of the LET rung itself.



Sequencer Initiation Rungs

2. When programming the LET rung, the *user-defined* block of registers that regulates the Sequencer must be entered correctly. Whatever register is specified to the left of the "=" sign contains the current step number. It is suggested that "ZZZZ" be equal to "XXXX" so that the Sequencer can be located in the program by SEARCHing for the first register in the block.

3. The rung containing the RESET/ENABLE bit should precede the LET rung. Other rungs can then be used to preset register values or control bits.

4. The following registers are part of the basic Sequencer set-up procedure and *must* be included in the Sequencer program:

   - **REGISTER D2, BITS 2 through 10.** Note the default here is all "0"s, meaning forward step/single-pass automatic operation.

   - **REGISTER D4.** Points to the register that is used to determine input conditions for advance. If sequence advance is based solely *on time, register D4 should equal* "0".

   - **REGISTER D5.** Points to the first output register written to by the step output states.

   - ***REGISTER D6.*** *Identifies the total number of* output bits occupying contiguous registers beginning with the register pointed at by register D5.

   - **REGISTERS D11** (1st step), **D17** (2nd step), **D23** (3rd step), etc. For each step, specifies the amount of time to be spent in the step prior to enabling a time-based advance. If the step advance is based *solely on input conditions*, these registers should all be zero.

   - **REGISTERS D12** (1st step), **D18** (2nd step), **D24** (3rd step), etc. For each step, specifies the binary pattern of "1"s that, when matched by the binary pattern in the input register pointed at by D4, enables a step advance based on inputs. If the step advance is based *solely on time*, the specified binary pattern in these registers should be all "0"s.

   - **REGISTERS 13 through 16** (1st step), **19** through **22** (2nd step), **25** through **28** (3rd step), etc. Output bit patterns that are user-*created for each step, in accordance with the* output field width indicated by register D6.

11.3.2 OPERATING CHARACTERISTICS

The Sequencer executes during each scan of the ladder program to assure program control of the outputs based on user-defined states. The Sequencer operates as follows:

1. When the Sequencer is RESET (bit 1 of the Operating Mode register = 0), all outputs in the program that are specified by registers D5 and D6 in the Sequencer are turned OFF. Current Step register (D1) is reset to zero. Current Step Running Time register D3 is also reset to zero.

2. When the Sequencer is ENABLED (bit 1 of Operating Mode register ="1"), the specified output states of step 1 (defined by registers D13 through D16) are transferred to the outputs defined by both the pointer register D5 and output width register D6. The Current Step Running Time register begins to accumulate time.

3. Step conditions for the next advance contained in registers D11 and D12 are compared with the real-time states of the Current Step Running Time register D3 and with the bit pattern of the input register being pointed at by register D4. Bit 10 of the Operating Mode register is checked to determine if either or both conditions must be met prior to a step advance. The Current Step Running Time register accumulates time as long as bit 3 of the operating mode register is not set to "1".

4. During every scan, in conjunction with the above comparisons, the Sequencer checks the status of bit 5 in the Operating Mode register D1. This is a check for AUTO or MANUAL operation. If MANUAL is set, the *time* and *input* advance parameters become meaningless since step advance is based solely on bit 7 being toggled from "0" to "1".

5. Whether in AUTO or MANUAL, the Sequencer now knows what criteria to monitor when determining a step advance.

6. If the Sequencer is in AUTO and the appropriate criteria for a step advance are satisfied, bit 4 of register D2 is checked to see if advance is inhibited.

7. If bit 4 = "0" (advance *not* inhibited), bits 8 and 9 of register D2 are checked to see if the advance is non-sequential.

8. If the advance *is* sequential, bit 2 of register D2 is checked to determine if the advance is forward (incrementing) or reverse (decrementing).

9. The Sequencer checks to see if it has just completed a pass through the final step. If it has, then bit 6 of register D2 is also checked.

10. Finally, if a step change is warranted, the output states for the appropriate step are written to the outputs being pointed at. The entire process then restarts.

The values of these Sequencer control registers and bits can be changed by the user at any time.

NOTE: *If the Current Step register (D1) is suddenly changed to a different valid step number, the new step is implemented the next time the Sequencer rung is scanned. New output states are written, Current Step Running Time is reset to "0". New Time Advance and/or Input Advance conditions associated with the new step apply.*

11.3.3 TYPICAL OPERATING MODES

This section presents insights into setting up the Sequencer for execution of some typical configurations. The focus here is on manipulating bits within the Operating Mode Register (Register D2, described in Figure 11.4).

As discussed in Section 11.3.1, registers 4, 5 and 6 of the *Configuration Table* indicate the Input Pointer, Output Pointer, and Output Width. These three registers, along with all the *Step Table* registers for Time Advance, Input Advance, and output states of each step, provide the necessary information for Sequencer operation. Normally, these registers once set, remain set in that pattern. However to guarantee the set pattern, it is advised to program additional LET rungs while configuring the Sequencer.

Although only two rungs are mandatory for Sequencer initiation (Figure 11.1), it is a good idea to use as many additional LET rungs in the program as practical. Using "LET presets" to set Sequencer values eliminates the chance of the values being lost, since the values are preserved in the ladder program.

A good practice is to locate the LET presets in the beginning of the program in conjunction with a GOTO that bypasses these statements after the Sequencer is initialized. This bypassing procedure eliminates the time penalty that redundant and unnecessary scanning of the LET rungs incurs.

Of course, the *Operating Mode register* bits can also be programmed into the ladder program. However, rungs containing these bits *must be executed every scan. To ensure proper set-up, the Operating Mode* register rungs should be located between the two mandatory rungs as shown in Figure 11.6.

The Sequencer is now assumed to be properly set up. Some common Sequencer operating modes are explained in the following and include TIME, EVENT, TIME/EVENT COMBINATION, and MANUAL sequencing.

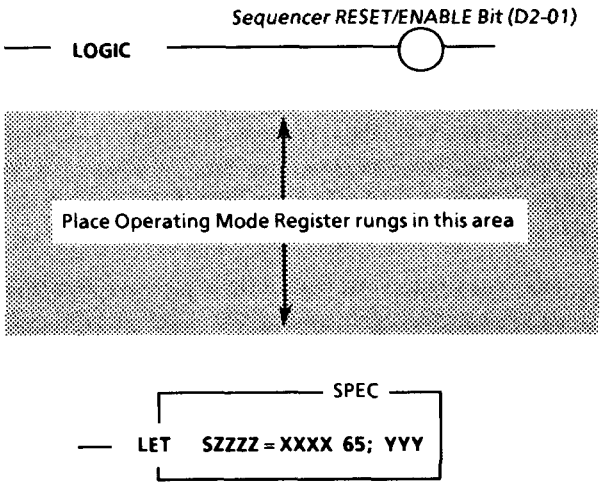


Figure 11.6 Operating Mode Rungs

TIME SEQUENCING ONLY

The following parameters apply for a time-advanced *Sequencer:*

- Operation is *automatic;* bit 5 of register D2 = "0".
- Since no Input Advance condition is used, set register D4 (and D12/D18/etc., depending on the number of steps) to "0".
- Enter the Time Advance condition into register D11 (and D17/D23/etc., depending on the number of steps). The time is entered in tenth-second (0.1 second) increments.
- Maximum time value for a single step is 32,767 (3,276 seconds, or 54 minutes and 36 seconds).

OPERATION: When bit 1 of register D2 is set to "1", the output pattern associated with Sequencer step #1 (registers D13 through D16) is written into the output register(s) being pointed at by registers D5 and D6 (*STEP #1 ACTIVE).*

When the time specified by register D11 has accumulated in register D3, the output pattern associated with Sequencer step #2 (registers D19 through D22) are written into the output registers being pointed at by registers D5 and D6 (*STEP #2 ACTIVE).*

Register D3 resets and begins timing again. When the time specified by register D17 has accumulated in D3, the advance to step #3 occurs (*STEP #3 ACTIVE).*

This process continues as long as bit 1 of register D2 is set to "1" and bit 5 of register D2 (AUTO/MANUAL select) is set to "0". When the last step in the Sequencer has been executed, bit 6 of register D2 (REPEAT/SINGLE PASS select) is checked to determine whether the Sequencer should "wrap around" to its step #1 or halt. If the Sequencer halts (bit 6 = "0"), all outputs are turned OFF.

POSSIBLE MODIFICATIONS: While the Time-Advanced Sequencer is running, the following modifications can be made:

- Instead of incrementing *forward*, the Sequencer can be set to operate in *reverse* (set bit 2 of D2 to "1").
- Upon step completion, *non-sequential* operation can be obtained (Bits 8 and 9 in register D2: bit 8 set to "1" executes the next step according to register D7, while bit 9 set to "1" non sequentially executes the next step in accordance to the contents in register D8). *Instantaneous* non-sequential operation can also be obtained by altering register D1.
- The *Current Step Running Time* can be held (inhibited from timing) by setting bit 3 of register D2 to "1".
- If the same output pattern must be maintained for more than the maximum time value (3,276 seconds), simply repeat the output pattern in multiple steps with appropriate Time Advance conditions.
- A *step advance inhibit* is enabled by setting bit 4 of register D2 to "1". This inhibits an advance even if the time requirement for the advance has elapsed.

INPUT SEQUENCING ONLY

The following parameters apply to an input-advanced Sequencer:

- Operation is *automatic;* bit 5 of register D2 = "0".
- Load register D4 with the address of the input register being pointed at. This is used for comparison to determine advance conditions.

  NOTE: *Register D4 is simply a pointer and is not the register that is actually compared.*

- All Time Condition advance registers (D11, D17, etc., depending on the number of steps being used) should be set to "0".
- Input Condition advance registers (D12, D18, etc., depending on the number of steps) should be preset with the bit pattern which, when compared to the register that's pointed at by D4, allows a step advance.

NOTE: *The bit pattern of "1"s and "0"s does not have to match exactly; it is only the "1"s in the Input Condition Advance register which have to be matched by the corresponding bits in the pointed to register. The "0"s are not matched. This means that if an advance is dependent upon one or more bits being "0", these bits need to be inverted prior to being compared.*

EXAMPLE: If the necessary condition for advancing to the next step is having bits 2 and 3 ON, then these two bits should be the only bits set in the Input Condition advance register. As soon as bits 2 and 3 are ON in the pointed-to register, a step advance is enabled *regardless* of how many other bits are ON in the register being pointed at.

OPERATION: When bit 1 of register D2 is set to "1", the output pattern associated with Sequencer step #1 (registers D13 through D16) is written into the output register(s) pointed at by registers D5 and D6 (*STEP #1 ACTIVE*).

When the bit pattern of "1"s specified in register D12 is matched by the pattern in the input register being pointed at by D4, the following occurs: The output pattern associated with Sequencer step #2 (registers D19 through D22) is written into the output register(s) being pointed at by D5 and D6 (*STEP #2 ACTIVE*).

Comparison for the third step advance is made with register D18. This process continues for as long as bit 1 of register D2 is set to "1" and bit 5 of register D2 (AUTO/MANUAL select) is set to "0".

After the last Sequencer step is executed, bit 6 of register D2 (REPEAT/SINGLE-PASS select) is checked to determine whether the Sequencer should "wrap around" to step #1 or halt. If the Sequencer halts (bit 6 = "0"), all outputs are turned OFF.

POSSIBLE MODIFICATIONS: While the Input Advanced Sequencer is running, the same modifications apply as for the Time Advanced Sequencer (Forward/Reverse stepping, non-sequential stepping, etc.). One exception is the *current step running time* parameter, which is not applicable in a Sequencer based solely on input conditions.

COMBINATION TIME AND EVENT/INPUT SEQUENCING

The following parameters apply to a Sequencer whose advances are based on *both* time and inputs:

- Operation is *automatic;* bit 5 of register D2 = "0".
- Register D4 should point to the register address whose bit pattern is used to compare for a step advance.

  NOTE: *Register D4 is simply a pointer and is not the register that is actually compared.*

- The Time Condition advance registers (D11, D17, etc., depending on the number of steps) should be loaded with times in tenth-second (0.1 second) increments.

  NOTE: *If a given step advance is to be based solely on input, "0" should be entered into the Time Condition register for that step.*

- The Input Condition advance registers (D12, D18, etc., depending on the number of steps) should be loaded with the bit patterns which, when compared to the register pointed at by D4, allow a step advance. As explained in the *INPUT SEQUENCING* section, it is only the "1"s in the Input-Condition advance register that are compared.

  NOTE: *If a given step advance is based solely on time conditions, "0" should be entered into the Input Condition register for that step.*

- If *both* time and event sequencing are specified in a step, an advance is not enabled *until the input condition has been met after the time condition is satisfied* (unless bit 10 of register D2="1"). This occurs even if the input condition has previously been met. If bit 10 of register D2 = "1", then a step advance is enabled upon *either* condition being satisfied.

OPERATION: When bit 1 of register D2 is set to "1", the output pattern associated with Sequencer step #1 (registers D13 through D16) is written into the output registers being pointed at by registers D5 and D6 (*STEP #1 ACTIVE).*

When register D3 is greater than or equal to D11 (i.e., the time condition is satisfied), the pattern of "1"s specified by register D12 is compared to the input register pointed at by register D4. When all the "1"s in D12 are matched by the register pointed at by D4, the output pattern associated with Sequencer step #2 (registers D19 through D22) is written into the output register(s) being pointed at by D5 and D6 (*STEP #2 ACTIVE).*

Register D3 resets and begins timing again. Comparison for the third step advance is made using registers D17 (for time) and D18 (for input). This process continues as long as bit 1 of register D2 is set to "1" and bit 5 of register D2 (AUTO/MANUAL select) is reset to "0".

After the last Sequencer step is executed, bit 6 of register D2 (REPEAT/SINGLE-PASS select) is checked to determine whether the Sequencer should "wrap around" to step #1 or halt. If the Sequencer halts (bit 6 = "0"), all outputs are turned OFF.

POSSIBLE MODIFICATIONS: While the combination Time and Input Advanced Sequencer is running, the same modifications apply as for the Time Advanced Sequencer (Forward/Reverse stepping, non-sequential stepping, etc.). One exception is the *current step running time* parameter, which is not used if a particular Sequencer step is based solely on input conditions.

MANUAL SEQUENCING

A manual Sequencer does not rely on time and/or input conditions to advance. Instead, advance is determined solely by the state of bit 7 in register D2, which is toggled manually.

NOTE: *If an input condition exists for advance, it is ignored. If a time condition exists, register D3 does not accumulate time. If desired, the input and/or time conditions for advance can be loaded in anticipation of automatic operation, because all input and time conditions are simply ignored in the manual Sequencer.*

The following parameters apply to a manual Sequencer:

- Operation is *manual;* bit 5 of register D2 is set to "1".
- The Sequencer is advanced by toggling bit 7 of register D2 from "0" to "1".

OPERATION: When bit 1 of register D2 is set to "1", the output pattern associated with Sequencer step #1 (registers D13 through D16) is written into the output registers being pointed at by D5 and D6 (*STEP #1 ACTIVE).*

If bit 5 of register D2 is set to "1", the next step advance occurs when bit 7 of register D2 is toggled from "0" to "1". At this time the output pattern associated with Sequencer step #2 (registers D19 through D22) is written to the output register(s) pointed at by D5 and D6.

NOTE: *To prepare for another step advance, bit 7 of register D2 must be reset to "0" before being toggled back to "1".*

Sequencer behavior can be set to automatic at any time by resetting bit 5 of register D2 to "0". After the last Sequencer step is executed, bit 6 of register D2 (REPEAT/SINGLE-PASS select) is checked to determine whether the Sequencer should "wrap around" to step #1 or halt. If the Sequencer halts (bit 6 = "0"), all outputs are turned OFF.

POSSIBLE MODIFICATIONS: While the manual Sequencer is running, the same modifications apply as for the automatic advance Sequencer (Forward/Reverse stepping, non-sequential stepping, etc.). Bit 4 of register D2 only applies to the AUTO mode operation and *cannot* be used to inhibit a MANUAL step advance.

NON-SEQUENTIAL STEPPING

Regardless of whether the Sequencer is operating in the manual or automatic mode, it is possible to alter the default incremental-forward stepping mode in the following ways:

1. Reverse advance (decrement step).
2. Non-sequential advance (i.e., jump to a step not immediately preceding or following the current step).
3. Immediate and unconditional non-sequential advance to another step, even though the advance conditions for the present step are not satisfied.
4. Inhibit a step advance while in the AUTO mode, causing the current step to be maintained despite any satisfied conditions for automatic advance that are present.

The non-default advance conditions are as follows:

Reverse (Decremental) Stepping. Set bit 2 of register D2 to "1", causing the Sequencer to step backward (reverse mode). Decrementing from Sequencer step #1 causes the Sequencer to go to step 0 (all outputs OFF) unless bit 6 of register D2 is set (REPEAT mode), in which case the Sequencer goes to the last step.

NOTE: *Bit 2 of register D2 has a lower priority than bits 4 (Step Inhibit), 8 or 9 (Non-Sequential Step) or setting Register D1 to the desired step.*

If the reverse mode is selected prior to Sequencer enabling, the Sequencer - upon initialization - jumps to the last step.

***Non-Sequential* Advance.** Set bit 8 of register D2 to "1", and load register D7 (first non-sequential step register) with the number of the next desired step, or set bit 9 of register D2 to "1" and load register D8 with the number of the next desired step. Bit 8/Register D7 and Bit 9/Register D8 function identically; two pairs are provided for user convenience.

NOTE: *If <u>both</u> register pairs are enabled simultaneously, Bit 8/Register D7 takes precedence.*

When either of these bit/register pairs is activated, the Sequencer goes to the step specified in either register D7 or D8 when the next step advance occurs. The number entered in these registers must be a valid step number (zero is NOT a valid step number); otherwise no advance takes place. Outputs remain as defined for the current step and bit 18 of register D2 is set to "1" to indicate an error.

If a non-sequential advance is specified prior to Sequencer enabling, the Sequencer advances - upon initialization - to the specified non-sequential step.

Immediate And Unconditional Non-Sequential Advance. Loading a step number into register D1 (Current Step Number) causes the Sequencer to jump unconditionally to that step upon the next program scan. The Sequencer detects this by storing the step executed during the previous scan in register D9.

At the beginning of each new scan, D9 is compared with D1. If they differ, the Sequencer immediately jumps to the new step specified in D1. Normal Sequencer operation resumes from that point – register D3 resets and begins timing, and the Input Condition advance register for the new step is used for comparison.

NOTE: *The new step that is jumped to must be valid for the Sequencer range defined. If the step is invalid, no step change occurs; outputs remain as defined for the previous step and bit 18 of register D2 is set to "1" to indicate the error.*

If an immediate and unconditional non-sequential advance is specified prior to Sequencer enabling, the Sequencer - upon initialization - advances to the first step for one scan before jumping to the selected step. Similarly, if a step advance is enabled while an immediate and unconditional non-sequential advance is specified, the advance goes active for one scan before reverting to the specified step.

Inhibit a Step Advance (AUTO mode only). Set bit 4 of register D2 to "1". This prevents the Sequencer from advancing beyond the current step, even if conditions for advance are satisfied when in the automatic mode.

The Automatic Step Advance Inhibit applies regardless of whether the step change is to be forward or reverse.

As soon as bit 4 of register D2 is reset to "0", output control reverts to control of the automatic advance condition. For example, if the only condition for advance is inputs (which happen to be satisfied), the step advance occurs the instant that bit 4 is reset to "0".

If a step advance is inhibited *prior* to Sequencer enabling, the Sequencer - upon initialization - advances to the *first* step. The *next* step advance is inhibited, provided the inhibit remains selected and the Sequencer is in AUTO mode.

Bit 4 only applies to automatic mode operation (bit 5 of register D2 = "0") and does not inhibit a manual or Immediate And Unconditional Non-sequential Step Advance.

11.4 Application Considerations

This section presents a collection of miscellaneous considerations for Sequencer operation.

11.4.1 ERROR CONDITIONS

Bit 18 of the Operating Mode register (register D2) is used to flag invalid Sequencer operating conditions. When an invalid operation is detected by the Sequencer, bit 18 is set to "1".

If the error is generated while the Sequencer is running, the condition of the previous step is retained. If the error exists prior to Sequencer enabling, no step execution occurs until the error is corrected.

The most common reasons for bit 18 to be set are listed below:

1. An undefined (non-existent) step has been specified in registers D1, D7 or D8.

   NOTE: *D7 or D8 are not checked until either bit 8 or bit 9 of the Operating Mode register = "1".*

2. The number of steps specified exceeds 255 (the Sequencer does not commence operation at all until this error is corrected).

3. An invalid Register Block starting address has been given in the Sequencer's configuration rung (Figure 11.1).

4. An invalid Output Width Definition has been entered into register D6 (width must be in the range 1 to 64).

5. Register D4 = "0", but a Step Input Advance condition register (D12, D18, etc.) exists and is not equal to "0".

When any of these error conditions appear, the Sequencer is reporting that it has been asked to perform an illegal operation.

When defining a register block for the Sequencer to use, make sure that sufficient registers exist between the start of the block and the last available register in the Model 650 (register 8000 is the last). Otherwise, the Sequencer flags an invalid operating condition and sets bit 18 of the register indicated to the left of the "=" sign.

To guarantee sufficient registers, apply the following equation (from Section 11.2):

Total Registers Per Sequencer
= [6x (# of steps) + 10]

Subtract the total registers per Sequencer from 8000 to obtain the *highest* address that the Sequencer can be started at.

EXAMPLE: A 50-step Sequencer is to be programmed into the Model 650. From the equation above, the registers required for this Sequencer are:

$$[6 \times (50) + 10] = \mathbf{310}$$

The highest address at which this Sequencer can be located is therefore:

$$8000 - 310 = \textbf{register \#7690}$$

The Sequencer's register block must begin at an address no higher than 7690.

NOTE: *Do not use registers allocated for a Sequencer for any other purpose.*

11.4.2 OPERATING MODE REGISTER STATUS BITS

Bits 17 through 23 of the Operating Mode register (D2) are read-only bits that are used to monitor the status of the Sequencer. The addresses of these bits can be programmed as contacts in the Model 650's user memory to provide detailed information on Sequencer operation. All of these status bits are reset to "0" when the Sequencer is reset (when bit 1 of register D2 is set to "0"). Each of the status bits is described below:

Bit 17 - Sequence Complete. When set, indicates that the Sequencer has just completed the final step. If bit 6 of register D2="0" (SINGLE-PASS SEQUENCE selected), bit 17 remains set at "1" until the Sequencer is reinitialized by toggling bit 1 of *register D2 from "0" to "1". If bit 6 of register D2 =* "1" (REPEAT SEQUENCE selected), bit 17 is set at "1" for only a single Model 650 scan as the Sequencer "wraps around" from the final step to the first step.

Bit 18 - Error Bit. This bit is described in Section 11.4.1.

Bit 19 - Manual Advance History. When set to "1", a manual advance has occurred. This bit is used to ensure that the manual advance bit (bit 7 of register D2) is transition-sensitive.

Bit 20 - Time Advance Enabled. When set to "1", the time advance condition for the current step has been satisfied.

Bit 21 - Input Advance Enabled. When set to "1", the input advance condition for the current step has been satisfied.

Bit 22 - Reset/Enable History. When set to "1", the Sequencer has been enabled. This bit is used to ensure that the reset/enable bit (bit 1 of register D2) is transition-sensitive.

Bit 23 - Step Advance Flag. When set to "1", a step advance has just occurred. Typically this bit is set for a single scan.

11.4.3 OTHER CONSIDERATIONS

Although *all* possible operating configurations cannot be examined, the following items should be considered when programming the Sequencer:

- OUTPUT STATES. Output states associated with a given step are written only if a step change occurs and are not refreshed each time the Sequencer is scanned. If the CPU is halted or power is removed when the Sequencer is in the middle of executing a step, the output states are not updated when programming execution resumes until a step change actually occurs. When the CPU resumes operation from a halt condition or power cycle, the external output registers are automatically reset. Sequencer *output states that are directly written to* external outputs remain OFF until the currently executing step has finished and a step advance has occurred. **If the user wishes to preserve the Sequencer output conditions for a given state following a processor halt or power cycle,** specify an internal storage register(s) in D5 and D6 and perform a data transfer from the storage register to the external output register.

- EXTERNAL OUTPUTS. Due to Image Table updating, output states that are associated with a given step are not written to external outputs *until the end of the Model 650's ladder scan.* The only exception to this is if the Immediate I/O Update function (bit 7 of 8176) is used.

- RUNG PROGRAMMING. It's recommended that any rungs which are used to modify the Sequencer's operation be programmed *prior* to the Sequencer initiation rung.

- SUBROUTINES. Since Sequencers are designed to be executed during each ladder scan, they should NOT be used in subroutines or in any part of a ladder diagram that could be skipped. Also, no logic should be placed in front of the SPEC 65 rung, to ensure the instruction is always executed.

- MCR. When preceded by a disabled MCR, the sequencer operation state is frozen: If reset, the Sequencer remains reset, while if in the middle of the step, the output of that step is maintained.

- All registers in the user-defined sequencer register block must have addresses in the range of 1 to 8000.

11.5 Examples

This section provides a simulated Sequencer to help clarify the Sequencer's set-up and operating procedures.

EXAMPLE SET-UP:

Assume the following Sequencer conditions for the example:

Number of Steps	20
Registers Required	130 = [(6×20) + 10]
Beginning Register Address	0751
Ending Register Address	0880
Output Register Address	0004
Output Width	12 bits (register #4, bits 1 through 12)
Input Register Address	0001 (any or all of the 16 bits can be used to establish advance conditions).

Upon Sequencer initialization (step #1), assume that bits 2, 3 and 4 of the output register (#4) are to be set to "1". The initiation rungs needed are shown in Figure 11.7.

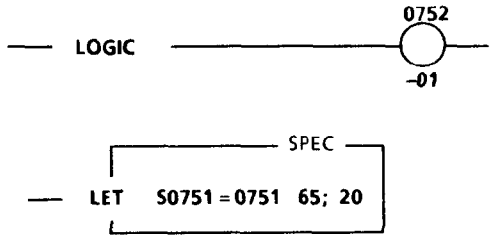


Figure 11.7 Example One Initiation Rungs

In the first rung shown in Figure 11.7, **0752-01** is the RESET/ENABLE bit for the Sequencer. In the second rung, **0751** is the address that identifies the start of the register block. **SPEC 65** identifies the special instruction as a Sequencer, while **20** is the number of steps in the Sequencer.

In accordance with Section 11.3.1, the following registers have to be set up for the first step (step #1) in Example One:

0752 (D2) .. Operating Mode (assume for now the default operation).

0754 (D4) .. Pointer to Input Register 0001. Preset to 1.

0755 (D5) .. Pointer to Output Register 0004. Preset to 4.

0756 (D6) .. Output Width Definition. Preset to 12.

0761 (D11) Time Condition Advance. Assume that bits 2 through 4 of register 0004 must be ON for 60 seconds to satisfy the time advance requirement. Preset to 600 (60 ÷ 0.1 sec. increments).

0762 (D12) . Input Condition Advance. Assume a step advance is enabled when bits 1, 3 and 4 of register 0001 are set to "1". Preset to 13 (decimal value. For corresponding binary pattern, see Figure 11.10).

0763 (D13) . Step #1 Output States bits 1 through 16. Since bits 2, 3 and 4 of register 0004 is set to "1", preset to 14 (decimal value). For the corresponding binary pattern see Figure 11.8.

0764 through 0766 (D14 through D16) ..... Step #1 Output States bits 17 through 64. Since the Sequencer's output width is less than 17, these registers are not applicable.

The remaining 19 steps in the Sequencer must be set up separately and are not shown here.

EXAMPLE OPERATION

Before bit 1 of register 0752 (D2) is energized, registers 0751 and 0753 equals zero and bits 1 through 12 of register 0004 is reset to "0" every time the initiation rungs (Figure 11.7) are scanned by the Model 650.

When bit 1 of register 0752 is set to "1", the following occurs:

1. The Sequencer sets register 0751 to "1".

2. (Also see "2A" below). Output information in registers 0763 through 0766 is transferred to the output registers specified by 0755 (D5) and 0756 (D6). Since D5 = 4 and D6 = 12, bits 1 through 12 of register 0763 are transferred to bits 1 through 12 of register 0004. Since register 0763 = 14, bits 2 through 4 of this register are set to "1" while bit 1 and bits 5 through 12 are set to "0" as shown in Figure 11.8 below:

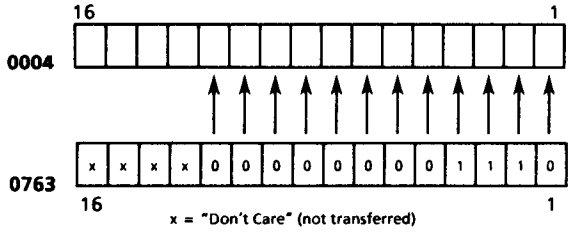


Figure 11.8 Transfer Of 12-Bit Output Width

2A. If the output width in this example was specified as *52* instead of 12 (register 0756 = 52), additional output registers would be used. As shown in Figure 11.10, all 16 bits of registers 0763, 0764 and 0765, along with bits 1 through 4 of register 0766 are written to registers 0004, 0005, 0006 and the first four bits of 0007. If the bit patterns of Figure 11.10 are involved in a transfer, it is necessary to preset register 0764 with the decimal value -24,562 (A00EH), register 0765 with decimal 16,391 (4007H), and register 0766 with decimal 15 (000FH). These settings allow the multi-register transfer to take place.

3. Bit 23 of register 0752 is set to "1" for one scan.

4. Register 0753 (D3) begins to time (to be compared with register 0761) to determine Time Advance enable.

5. The bit pattern of register 0001 (the Input Register pointed at by register 0754) is compared to register 0762 to see if the condition for Input Advance is satisfied. The Input Advance does not enable a step advance until the time condition (register 0753 = register 0761) has been satisfied.

6. Step advance to step #1 is now complete. Assuming no Operating Mode register bits are set, the next step advance occurs when bits 1, 3 and 4 (specified in register 0762) of register 0001 (pointed at by register 0754) are set to "1" as shown in Figure 11.9 below.

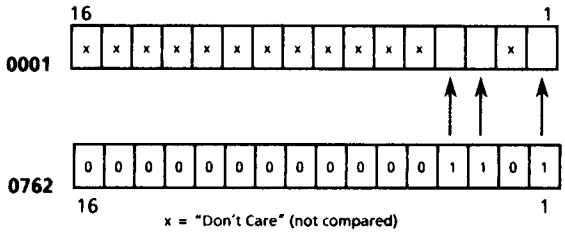


Figure 11.9 Input Advance Condition Satisfied

Before this comparison occurs, the time condition for advance (60 seconds) must be satisfied.

NOTE: *If it is desired to have the step advance in this example based solely on time, presetting register 0762 to zero allows the advance to occur after 60 seconds. If a strictly input based advance is desired, presetting register 0761 to zero allows an advance as soon as bits 1, 3 and 4 of register 0001 are set to "1".*

The following is a list of options that are not necessary for basic Sequencer operation. The description of Sequencer operation continues (at step #7) after the options.

OPTIONS:

Now that the Sequencer is enabled, a variety of operating modes can be selected by altering the bits in the Operating Mode register (register 0752). Some possibilities are listed below:

Reverse Sequencing- Set bit 2 of register 0752 to "1", and the Sequencer runs in reverse. If the Sequencer is currently at step #1, enabling bit 2 advances the Sequencer to step #20 (the last step) instead of to step #2.

Prevent Time Accumulation - Set bit 3 of register 0752 to "1" and no time accumulates in the Current Step Running Time register (0753).

Inhibit A Step Advance - Set bit 4 of register 0752 to "1" and the Sequencer does not advance from the current step – even if time (register 753 = 600) and/or input (bits 1, 3 and 4 of register 0001 = "1") conditions are satisfied. Also, output register 0004 does not change.

Manual Operation - Set bit 5 of register 0752 to "1" and the Sequencer ignores both time and input advance conditions. Toggle bit 7 of register 0752 to manually step the Sequencer.

Non-Sequential Stepping - Bits 8 and 9 of register 0752 are used to make the Sequencer jump to a step that is not in numerical order (for example, jump from step #1 to step #4, ignoring #2 and #3). Register 0757 or 0758 needs to be loaded with a valid step number (1 to 20), otherwise the current step is maintained. If bit 8 or 9 of 0752 is set, and advance conditions are satisfied, the Sequencer jumps to the step specified in register 0757 (if bit 8 is set) or register 0758 (if bit 9 is set).

NOTE: *If an immediate jump to a non-sequential step is desired, load the valid step number (1-20) into register 0751. Step advance occurs even if time and/or input conditions are not satisfied.*

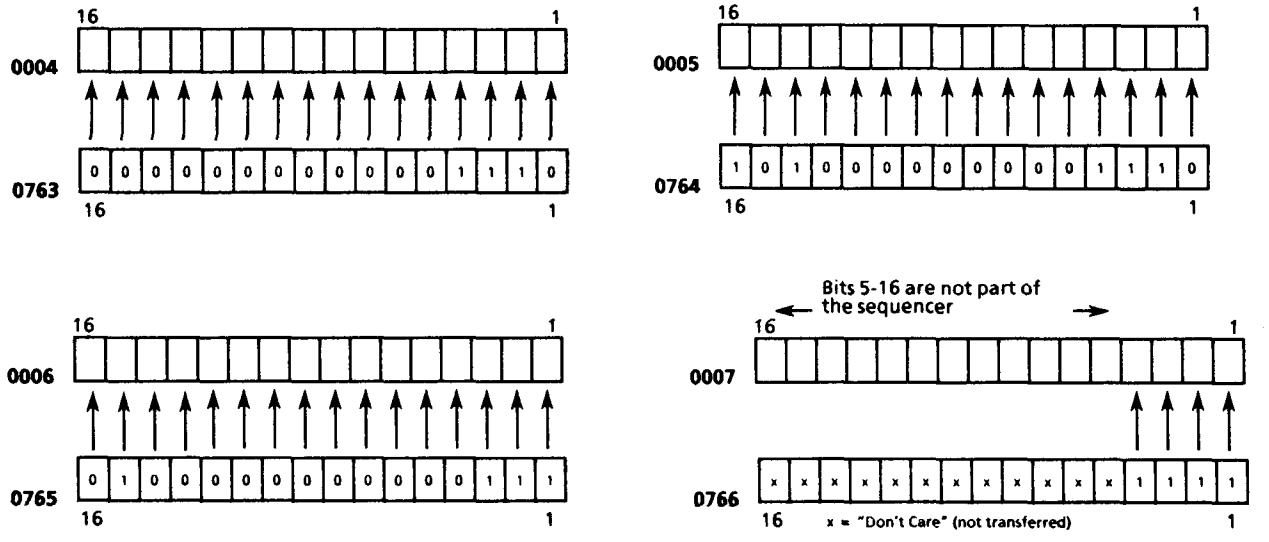


Figure 11.10 Transfer Of 52-Bit Output Width

Time or Input Condition Advance - Set bit 10 of register 0752 to "1" and a step advance is enabled as soon as *either* register 0753 = 600 (60 seconds) *or* bits 1, 3 and 4 of register 0001 are set to "1".

7. Sequencer step #2. Assume the initial conditions of this example are unchanged and that no Operating Mode register bits are set to "1". Fix the output bit pattern for step #2 such that bits 1, 4 and 5 of register 0004 are to be set to "1" (all other bits "0"). This is accomplished by setting register 0769 to 25 (0019H). Figures 11.11A and 11.11B are the requirements and results of Sequencer step #2.

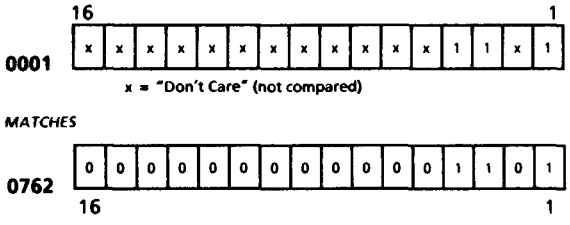


Figure 11.11A Step #2 Advance

When register 0001 matches register 0762 (as shown in Figure 11.11A), and register 0753 ≥ register 0761 (step #1 has been running for at least 60 seconds), the following occurs:

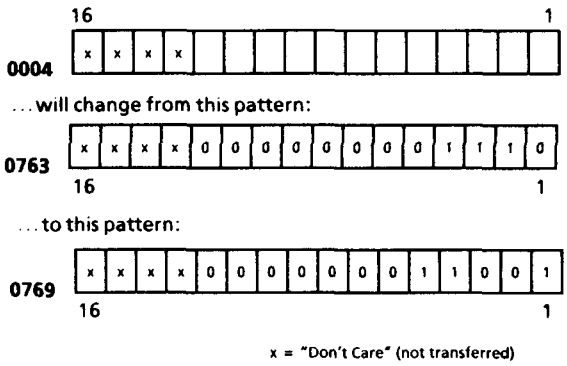


Figure 11.11B Step #2 Result

For *manual* operation (bit 5 of register 0752 = "1"), register 0004's pattern would change from the pattern in register 0763 to the pattern of 0769 when bit 7 of 0752 is toggled from "0" to "1".

This procedure is followed for each step as long as the Sequencer is enabled. If the Sequencer is disabled by resetting bit 1 of register 0752, the outputs are turned OFF and the current step register (0751) is reset to 0. Upon reaching the last step (step 20), bit 6 of register 0752 is monitored. If bit 6 is set, the next step advance is to step 1 and the Sequencer operation continues. If bit 6 is reset, the next step advance concludes the sequencing operation and all outputs are turned OFF.

12 BATTERY BACKUP AND CLOCK/CALENDAR

12.1 Backup Battery

12.1.1 DESCRIPTION

The Model 650 uses two methods to supply backup power to the onboard volatile RAM and real-time clock/calendar circuitry in the event that power to the power supply is interrupted:

1. Batteries located in the SY/MAX power supply (PS-21, 31, etc) are the primary source of backup power for the Model 650 RAM and real-time clock/calendar in the event that +5VDC from the power supply is lost. Refer to Instruction Bulletin 30598-156-XX or 30598-159-XX for information on the power supply batteries.

2. The Model 650's onboard lithium battery supplies power to the RAM and real-time clock/calendar if the voltage from the batteries in the power supply is not sufficient to retain valid data in the RAM. The battery also maintains the RAM contents and clock/calendar accuracy if the processor is removed from the rack or shipped to use in another location. All 8192 Control Processor registers are battery-backed. Ethernet NIM registers 1 to 10 and 3000 through 7999 are battery-backed.

12.1.2 LOW BATTERY INDICATION

The condition of either the SY/MAX power supply batteries or the Model 650's onboard lithium battery is indicated with the BATTERY LOW LED on the front panel of the processor. See Section 4.2 for a complete explanation of this LED. Low battery condition may also be monitored as the status of a contact (bit 17 of register 8176 for the power supply batteries, bit 32 of register 8176 for the on-board lithium battery). Refer to Section 13, p. 9-11 for more information on Register 8176.

NOTE: *If the onboard battery is not installed, the BATTERY LOW indicator signals a low internal battery condition.*

12.1.3 BATTERY LIFE EXPECTANCY

The life expectancy of the onboard lithium battery is dependent on the time that the battery must supply current to the Model 650, and on the physical age of the battery.

Figure 12.1 gives a rough estimate of how long the Model 650's RAM is supported when a battery ranging from new to five years old is installed in a Model 650. The BATTERY AGE assumes the battery has been installed in the Model 650 but *not supplying current*, and that the Model 650 has been at its maximum operating temperature of 60 °C. The RAM DATA RETENTION TIME is based on a Model 650 at 40 °C, since the processor is not at operating temperatures while on lithium battery backup.

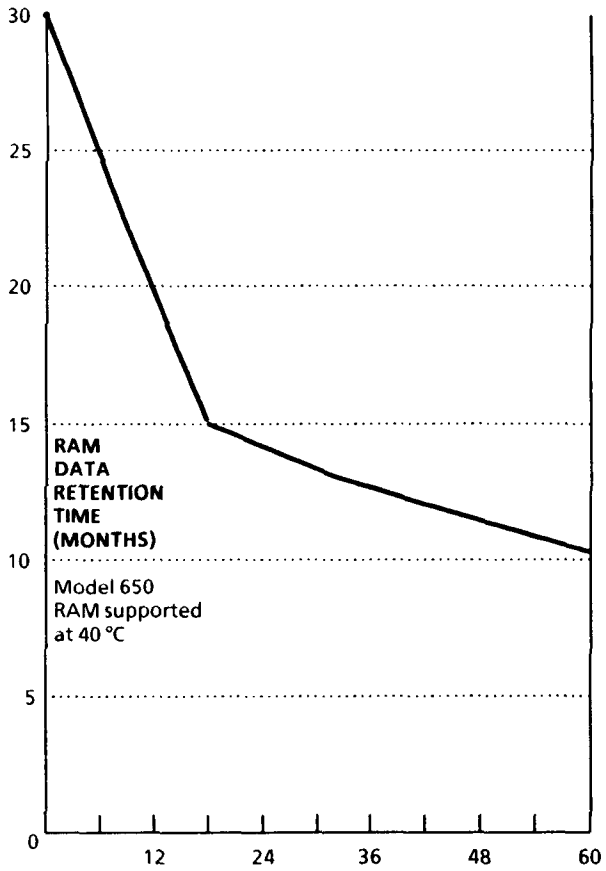


Figure 12.1 Backup Lithium Battery Life

12.1.4 REPLACEMENT PROCEDURE

The internal lithium battery is replaced by unscrewing the small access cover on the front panel of the Model 650. The battery can be changed while the Model 650 is operating. If the processor is installed in a rack which is supplying either regular or battery power, there is an unlimited time in which to replace the battery.

NOTE: *For maximum onboard battery life, the Model 650 should remain in the rack, connected to the power supply, as much as possible. In this way the power supply batteries – not the internal battery – are supplying any needed backup power.*

12.1.5 BATTERY SPECIFICATIONS

Square D Part Number 29904-08961

Manufacturer ..... Tadiran Electronic Industries Inc.

Mfg. Type Number TL-5104

Mfg. Catalog Number 15-51-04-210-000

Rated Voltage ..... Lithium AA, 3.5 to 3.6 V

Capacity (200 uA @ 3V) ... 1.9 Amp Hours (Typical cells stored at 25 °C for one year).

(alternate manufacturer source is SAFT, battery type LS6, catalog number 500200)

12.2 Real-Time Clock/Calendar

The Model 650 is equipped with a battery-backed real-time clock/calendar. The clock/calendar keeps track of seconds, minutes, hours, day of the week, day of the month, month, and year and clock compensates for leap years.

Values for the real-time clock/calendar reside in registers 8109 through 8115. These registers are updated in the Model 650 image table each hour by the regulation circuitry within the real-time clock. In between the hourly updates, the clock is updated second-by-second by the software oscillator in the Model 650.

12.2.1 SPECIFICATIONS

Format ..... 24-hour ("military") time. If 12-hour (AM/PM) mode is desired, user program must perform the conversion.

Accuracy ..... Less than one second per day @ 25 °C, 16 seconds per day over the 32 to 140 °F (0 to 60 °C) range.

Set Method ..... Write to registers using one of the following methods:

- Model 650 user program
- Programming equipment
- Other processors
- D-LOG Module

12.2.2 LOADING THE TIME/DATE

Model 650 registers 8108 through 8115 are assigned to the clock/calendar. Figure 12.2 shows the function of each of these registers:

REGISTER	FUNCTION	DECIMAL RANGE
8108	Set clock (see Section 12.2.4)	-1, 0, or +1
8109	Seconds	0–59
8110	Minutes	0–59
8111	Hours	0–23
8112	Day of week	1–7 (1 = Sunday)
8113	Day of month	1–31
8114	Month	1–12
8115	Year	1980 - 2079

Figure 12.2 Clock/Calendar Registers

Set the current time of day in the clock by the following procedure:

1. Using a CRT Programmer or equivalent, load *"–1" into register 8108. This interrupts the* updating of the image table clock registers; it does not affect the real-time clock itself.

2. Access each of the registers shown in Figure 12.2 by *starting with register 8109*. Register 8108 is only used to transfer data to the clock.

3. While in DATA ENTER mode, load each of the registers 8109-8115 with the appropriate value. Since these are *control* registers, it is necessary to strike the ENTER key multiple times to load the values.

EXAMPLE: To set the current time as 3:10 PM on Friday, February 26th, 1988, the following values would be entered into registers 8109--8115:

8109	00
8110	10
8111	15
8112	6
8113	26
8114	2
8115	1988

12.2.3 SETTING THE CLOCK

After the loading procedure in Section 12.2.2 has been completed, the values in the image table must be transferred to the clock. This is accomplished by loading the decimal value **0** into register 8108. The instant that this value is loaded into register 8108, the clock resumes keeping time using the loaded values.

TO UPDATE AND RESTART THE CLOCK:

8108	0

12.2.4 CLOCK MANIPULATION

NORMAL OPERATION (Register 8108 set to 0)

When manipulating the clock, it's important to realize that the clock circuitry itself is running regardless of whether the Model 650 is powered up or *not*.

During "normal" clock operation, the user-accessed image table registers are updated directly from the clock once each hour. During each hour, the Model 650's oscillator regulates the updating of the image table each second. The specified drift of the software oscillator while the Model 650 is powered up is ±1.8 seconds per hour. This drift is eliminated each time the real-time clock performs its hourly updates on the image table registers.

SET CLOCK OPERATION (Register 8108 set to –1)

To set the clock either via DATA ENTER mode or an instruction in the user program, it is first necessary to set register 8108 to "–1". This prevents the image table from being updated by *either* the real-time clock or by the Model 650's software oscillator.

NOTE: *The clock itself continues to run internally when register 8108 is set to "–1".*

After the desired values have been loaded into registers 8109-8115 (see Section 12.2.2), reset register 8108 to "0" (see Section 12.2.3). This action transfers the values in 8109-8115 to the real-time clock itself and normal operation resumes (i.e., the image table registers are updated each second by the software oscillator and each hour by the real-time clock).

NOTE: *If out-of-range values are entered into any register, the values are rejected and an ERROR 42 is displayed on the programmer.*

STOPWATCH OPERATION (Register 8108 set to + 1)

A special operational mode is enabled when register 8108 is set to "+1". In this "stopwatch" mode, the hourly updates of the image table by the real-time clock are inhibited, but the second-by-second updates by the software oscillator continue. When register 8108 is reset to "0", normal clock operation resumes and the image table registers are updated with the most current hourly time in the real-time clock.

This stopwatch mode can be very useful for timing discrete events. Since the software oscillator continues to run, time can be accumulated without being overwritten by an update from the real-time clock. At the conclusion of the timed event, resetting register 8108 to "0" resumes normal operation and registers 8109 to 8115 are updated automatically by the real-time clock. There is no need to reset the time of day since this is retained in the real-time clock during the event's timing.

The three modes of operation are summarized below in Figure 12.3.

REGISTER 8108 Set To:	Clock Mode	External Clock Registers Being Updated By:	
		SOFTWARE	REAL-TIME CLOCK
-1	Set Clock	NO	NO
0	Normal	YES	YES
+1	Stopwatch	YES	NO

Figure 12.3 Clock Operation Modes

In summary, when register 8108 is set to "-1" external clock registers are not updated by either the real-time clock or the software oscillator. This allows a user to preset the clock registers for the instant when register 8108 is user-reset to "0". The preset values are simultaneously transferred to the real-time clock at this point. If register 8108 is set to "+1", the clock registers operate in the "stopwatch" mode.

13 PROCESSOR CONTROL REGISTERS

13.1 Control Register Overview

NOTE: *Refer to Section 15.5.3 for an overview of Ethernet NIM (Control) Registers.*

The Model 650 Control Processor uses a total of 8192 register addresses. These processor registers are separated into the groups illustrated in Figure 1.1.

Control registers are reserved for Model 650 control functions. Register addresses 8093-8192 are reserved for these control functions within the control processor.

A register consists of 32 bits. Bits 1 through 16 are referred to as the *data field*, while bits 17 through 32 are referred to as the *status field*. The status field is unalterable, but may contain useful information about how the register is being used. The structure of a data register is shown in Figure 13.1.

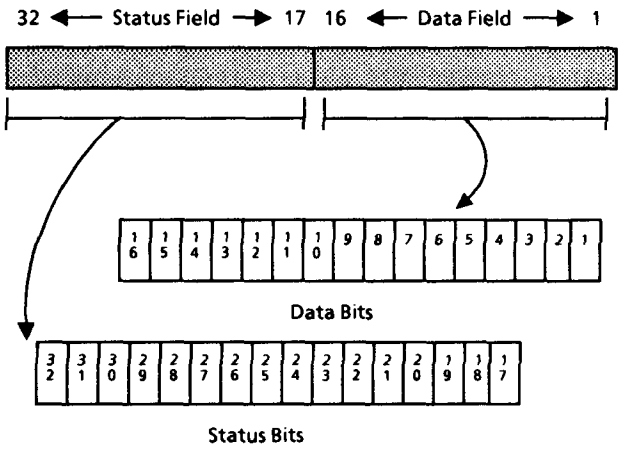


Figure 13.1 Register Structure

The control register addresses can be divided into two groups as follows:

Group #1 consists of addresses 8093-8178. Both the data field and status field of group #1 can be examined by the user, however only the data field can be altered.

Group #2 consists of registers 8179-8192. This group is used by the processor for certain "bookkeeping" functions. Only the data field is accessible to the user and can only be examined, not altered.

The examination of control registers and their individual bits can be accomplished by viewing them on a programmer or by using them as contacts within the ladder program.

WARNING

Altering control register bits with a programming device may affect actual equipment under control of the programmable control system. Make sure that all effects of altering control bits are understood prior to their alteration. An incorrect control register bit setting may result in equipment damage or bodily injury.

If control register bits are to be altered, the DATA mode of a programmer can be used or the bit can be programmed in the ladder as a coil.

13.2 Listing of Registers

8001–8005: Command identifiers (not actual storage registers) which identify the ASCII output format.

These registers are used as commands to set the format of the Model 650's output data when the Model 650 is executing a PRINT statement and are not intended as usable storage registers for other purposes. Available formats include:

- ASCII in decimal, hexadecimal, or binary
- "Raw binary" data
- Repeated character strings

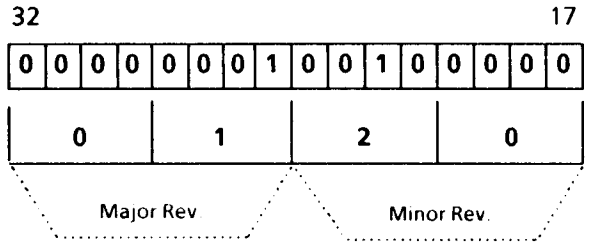
Refer to the programming device's instruction bulletin for programming information and Section 10.2 for application information.

8006: Acts as the command identifier (not an actual storage register) for the ASCII input format. See Section 10.

8007-8092: Not available.

GROUP #1: Read/Write Registers

8093: The 16-bit data field stores the SY/MAX drop number on Ethernet indicated by the rotary and DIP switches. Refer to Section 15.5.3 for Ethernet NIM Register 3 (similar to register 8093). The 16-bit Status field stores the version for the Ethernet communications software. Status field information is stored in a Binary Coded Decimal (BCD) format. The first two BCD digits (most significant byte) show the major software revision, followed by an implied decimal point. The last two BCD digits (least significant byte) indicate the minor revision of the software. For example:



Register 8093 Status field

Register 8093's status field indicates a major software revision of 1 and a minor revision of 20. This is read as revision 1.20 of the Ethernet communications software.

8094: Ethernet Port Comms Error. In those cases where register 8175 contains error code 929, register 8175 points to register 8094. Register 8094 contains additional information about the nature of the Ethernet Port Comms error. (See Section 15.7).

8095: Acts as a pointer for the indirect ROUTE LIST using the Ethernet (Channel 3) port. See Section 5.5.

8096: Not used.

8097: Acts as a pointer for the indirect ROUTE list using the PRGMR (channel 1) port. See Section 5.5.

8098: Same function as 8097, except controls COMM (channel 2) port.

8099: PRGMR port (channel 1) attributes. Refer to the following tables and Section 10 for bit explanations of this register.

8100: COMM (channel 2) attributes - same bit patterns as 8099.

8101 to **8104:** Not used.

8099 Bits 1-4: Baud Rate:

BIT PATTERN	BAUD RATE
0000	50
0001	110
0010	300
0011	1200
0100	2400
0101	4800
0110	9600
0111	19200

8099 Bits 5-6: Word Size:

BIT PATTERN	WORD SIZE
00	8 bit
01	7 bit
10	6 bit
11	5 bit

8099 Bits 7-8: Parity:

BIT PATTERN	PARITY
00	Even
01	Odd
10	None
11	None

8099 Bits 9-10: Stop Bits:

BIT PATTERN	STOP BITS
00	1
01	1.5
10	2
11	2

8099 Bit 11: ASCII Output Control Function (see Section 10.2.5)

BIT PATTERN	FUNCTION
0	XON/XOFF Disabled
1	XON/XOFF Enabled

8099 Bit 12: ASCII Input Null (00) Characters

BIT PATTERN	FUNCTION
0	Null ignored
1	Null stored

8105: Pointer to the first register for Immediate I/O Update instruction. See bit 7 of register 8176 later in this section.

8099 Bit 13: Transmit Periodic DLE/ENQ Stream (SY/MAX Protocol-Handshaking Characters)

BIT PATTERN	FUNCTION
0	Suppressed
1	Active

8099 Bit 14: Mask 8th Bit of ASCII Output Word (Reset Model 650's 8th ASCII output bit to "0")

BIT PATTERN	FUNCTION
0	No mask
1	Mask 8th bit

8099 Bit 15: CTRL-C Flag

BIT PATTERN	FUNCTION
0	Not Received
1	Received

8099 Bit 16: Clear Queued ASCII Input Buffers

BIT PATTERN	FUNCTION
0	Inactive
1	Active

8106: Register block size for the Immediate I/O Update instruction. See bit 7 of register 8176 later in this section.

8107: Pointer to the Error Register Stack (16 register block). Can be set by the user to point to the beginning of a block of 16 registers that preserve error codes appearing in register 8175. Reserve this 16-register block--which can be assigned to any register between 1 and 7985 --exclusively for use as the Error Register block.

In normal operation, error codes appearing in register 8175 are zeroed when the fault is cleared. Register 8107 provides a valuable troubleshooting aid by allowing the user to create a history of error codes.

As each new error is received, it is inserted into the first register position in the 16-register stack. Errors already in the stack are pushed down one position. When the stack is full, the oldest error is pushed out of the stack and is lost.

For example, if a particular error code occurs more than once consecutively (i.e., if ERROR 20001 enters the stack four consecutive times), it only appears once in the stack. Thus, multiple failures may only produce one error code in the stack if the failures are all caused by the same error code condition.

8108: Clock - Command Register.

8109: Clock - seconds (0-59).

8110: Clock - minutes (0-59).

8111: Clock - hours (0-23).

8112: Clock - day of week (1-7, 1 = Sunday).

8113: Clock - day of month (1-31).

8114: Clock - month (1-12).

8115: Clock - year (1980-2079).

8116-8118: Not used.

8119: Last register defined for channel 2 of the 15th LI module.

8120–8163: For use in communicating with Local Interface modules. Each 2-channel LI module utilizes up to 3 processor control registers for monitoring I/O and system control communications. Registers 8161-8163 of this group are reserved for the first LI module as explained in the following:

(8161) Control Register for channel 2 of the first Local Interface module, is used to monitor and control the conditions of the remote racks that contain the register modules and I/O connected to channel 2 of this local interface. The bit pattern of 8161 is as follows:

Bits 1-8: The HALT/RUN bit can be set to force the corresponding drop to shut down.

Bit 9: The FREEZE/RESET bit allows the user to choose if the channel should freeze (remain in last previous state) or reset (turn OFF) when a communication error exists. 1 = FREEZE, 0 = RESET.

Bit 10: XMIT FAULT TOLERANCE bit. When set to "1", allows up to 10 attempts to re-establish communication prior to generating an error. When this bit is reset to "0", only three attempts are allowed.

Bit 13: The FAILURE OVERRIDE bit allows the system to continue operation should communication to the remote I/O racks fail. 1 = Failure Override Mode 0 = Normal Operation.

Bit 15: AUTO RESTART bit. When set, automatically attempts to re-establish communication should an I/O fault occur on the channel. 1 = Auto Restart ON; 0 = Auto Restart OFF. Must be set in conjunction with bit 13 to be practical.

NOTE: *The remainder of the bits in register 8161's data field are unused.*

(8162) Channel 1 of the first Local Interface module. (This register performs the same function as register 8161 does except it monitors and controls Channel 1).

(8163) Error code control register for the first local interface. Refer to Appendix A for the error code table.

8164: A 2.5 ms "tick" register. Increments every 2.5 milliseconds regardless of the RUN/HALT status of the Model 650. All 32 bits of the register are used to keep track of accrued time. Maximum accrued time is 4 months. This register is reset upon its "rollover" or power up and can be preset by a programmer or from the ladder program. Whenever the data field of register 8164 is reset to zero, the status field also reverts to zero.

If the contents of register 8164 are used in an IF or LET rung, register 8164 should be "ORed" with zero before performing the data compare (IF) or transfer (LET). This effectively masks off the 16 bits of the status field, and prevents an overflow condition from occurring if any of these bits are non-zero.

EXAMPLE: LET S3995=S8164 executes as expected until register 8164 exceeds +32,767, at which time the contents of register 3995 stops changing. Bit 18 (overflow) of register 3995 is set to indicate the overflow condition. If register 3995 should remain "free-running", use the following rung:

LET S3995 = S8164 ∨ 0

After reaching +32,767, the contents of register 3995 changes to -32,768 then decrements back to zero.

8165: Used to establish safeguard rung parameters (see Section 6.8). Also contains the number of "ticks" remaining before the next timed interrupt is due to be called.

8166: Programmable minimum scan-time register. A positive number entered into this register establishes the minimum CPU ladder program scan time, to a resolution of 2.5 milliseconds. The Model 650 executes the user program followed by a series of "idle" instructions until the minimum scan time has elapsed. The desired scan time can be entered in 2.5 ms increments that are rounded up. For example, entering a minimum scan time of 98, 99 or 100 ms results in a ladder program scan time of not less than 100 ms and not more than 102.5 ms. Any scan time entered is subject to being increased by the I/O update time, plus the 5 ms port servicing delay.

8167: This register contains the programmable scan limit when the special safeguard rung (TLET S8165) is not used. When the scan time in milliseconds exceeds the contents of this register, the processor halts and generates ERROR 970. Maximum allowable scan time is 32.765 seconds. If this register contains zero, the scan time is limited to one second. If this register contains a value larger than 1 second (1000), it automatically is reset to zero (i.e., 1 second) upon power up. Accuracy is within 2.5 milliseconds, rounded up.

8168: Communications monitor timeout. This register defines the time allocated for receiving a reply to a READ, WRITE, or ALARM rung before declaring a message timeout and setting bit 23 of the communication status register. Bits 1-4 are used with the PRGMR (Channel 1) port, bits 5-8 with the COMM (Channel 2) port, and bits 9-12 with the ETHERNET (Channel 3) port.

The following table identifies the time interval range for bits in the 8168 register and is valid for all ports:

8168 Bits 1-4, 5-8 and 9-12:

BIT PATTERN (bits 1-4 for PRGMR bits 5-8 for COMM bits 9-12 for ETHERNET)	TIME COUNT IN MILLISECONDS
0000	0 to 250
0001	250 to 500
0010	500 to 750
0011	750 to 1000
0100	1000 to 1250
0101	1250 to 1500
0110	1500 to 1750
0111	1750 to 2000
1000	0 to 2000
1001	2000 to 4000
1010	4000 to 6000
1011	6000 to 8000
1100	8000 to 10000
1101	10000 to 12000
1110	12000 to 14000
1111	14000 to 16000

8169: This register assigns the BAUD rate for the Model 650's two communication ports. The first half of this register's data field is divided into two groups of 4 bits each. Each group of bits contains the baud rate for an individual port (channel).

The following table lists the channel-to-bit association for the PRGMR (Channel 1) port and the COMM (Channel 2) port.

8169 Bits 1-4 and 5-8

BIT PATTERN (bits 1-4 for PRGMR, bits 5-8 for COMM)	BAUD RATE
0000	50
0001	110
0010	300
0011	1200
0100	2400
0101	4800
0110	9600 (default)
0111	19200

8170:

Bits 1 to 15: Not used.

Bit 16: Activates the communication completion feature of the Model 650. (Also called the task swap enable/disable bit.) If this bit is set to 1 by the user, End of Scan communication servicing could be extended an additional 5 msec to allow for complete message servicing. See Sections 5.6.1 and 14.2.2.

8171: Contains the remainder of the most recent integer division performed. Remainder's sign (±) agrees with the numerator.

8172: Processor scan time in milliseconds. Scan time is the length of time required for the Model 650 to make one pass through user memory and includes end-of-scan I/O update and communication port servicing times.

8173: Fenced registers ending address (as entered by user). The *ending* point of the block of registers accessible via the COMM port (Channel 2).

8174: Fenced registers beginning address (as entered by the user). The *beginning* point of the block of registers accessible via the COMM port. Refer to Section 6.13 (security) for more details on the use of fencing.

8175: Error code storage. Refer to Appendix A for descriptions of these codes.

This register contains the last ERROR NUMBER which occurred. If there is more than one error, the most serious or highest priority error code is stored. The contents of register 8175 are cleared after a HALT-to-RUN keyswitch toggle, provided the cause of the error was eliminated. An exception to this are errors in serial communication, which clear as soon as a successful communication message has been exchanged through the indicated port. Errors occurring in register 8175 can be preserved in an "error register stack" using register 8107.

8176: Primary Processor Control Register. The following table explains each bit in this register.

The individual bits of register 8176 are used for system monitoring and control. Because it is possible to alter the state of the data field bits, the user has the capability to change specified processor conditions. These bits can be altered via the CRT programmer or SFW Software's DATA ENTER mode or by a ladder rung in memory. Bits 1 through 16 are alterable. Exceptions to this involve the use of certain memory write protection modes. Refer to Section 6, "Software Security".

REGISTER 8176 READ/WRITE BITS - The following bits can be altered by the user:

Bit 1: When this bit is set ON, it prevents the Model 650 from scanning the ladder program. The processor can be placed in the HALT mode by either putting the processor keyswitch in the HALT position or by programming and energizing coil 8176-1 in the ladder diagram program. Toggling the keyswitch into and out of HALT or setting and resetting 8176-3 resets this bit to OFF.

Bit 2: When this bit is set ON, the processor operates in the DISABLE OUTPUTS mode. The processor can be placed in the DISABLE OUTPUTS mode either by putting the keyswitch in the DISABLE OUTPUTS position, or by programming and energizing coil 8176-2 in the ladder diagram program.

Bit 3: This bit operates as a remote HALT/RUN switch. 1 = Processor Halt, 0 = Processor Run. The *HALT condition overrides the* RUN condition. Putting the keyswitch in HALT or setting bit 3 to "1" halts the processor. To put the processor in RUN, however, the keyswitch must be in RUN or RUN/PROGRAM and bit 3 must be "0".

NOTE: ***Performing a DEL/CLR ALL operation does NOT reset bit 3; once set, it must be individually reset by the user.***

8176 Bits 1-32

BIT	USED FOR:
1	Processor HALT
2	*Processor Disable Outputs*
3	Processor HALT/RUN
4	Memory Protection
5	Force Inhibit
6	Register Protection
7	Immediate I/O Update
8	IF Rung Overflow Flag
9	Control Register Data Corruption Flag
10	MATRIX or ARRAY Overflow Flag
11	Immediate Communications Update, Channel 1
12	Immediate Communications Update, Channel 2
13	Immediate Communications Update, Channel 3
14	**Not used**
15	Port and Route Edit Lockout Control
16	Inhibit Coil Address
17	Either Internal or Power Supply Battery is Low
18	OPEN Contact Address
19	SHORT Contact Address
20	**Not used**
21	First Dummy Scan
22	Second Dummy Scan
23	Normal Program Scan
24-31	**Not used**
32	Internal Battery Low

Bit 4: Setting this bit to "1" protects user memory from being altered via the COMM port (Channel 2) and Ethernet Port (Channel 3). For further details refer to Section 6.10.

1 = Memory Protected
0 = Memory Accessible.

Bit 5: Setting this bit to "1" inhibits the forcing of any I/O via the COMM port and Ethernet port (Channel 3). 1 = Force Inhibit, 0 = Forcing Allowed. I/O can still be forced *using programming equipment through the* PRGMR port (channel 1). Refer to Section 6.11.

Bit 6: In addition to the fencing capabilities of the Model 650, setting this bit protects any registers from being altered via the COMM port and Ethernet port (Channel 3). Registers can still be forced or altered through the PRGMR port using programming equipment. 1 = register protect, 0 = register accessible. Refer to Section 6.12.

Bit 7: Activates the Immediate I/O Update feature of the Model 650. When the Model 650 encounters a rung containing an energized bit 8176-7, it momentarily stops scanning the ladder program and updates the external I/O that are defined by registers 8105 and 8106 (inputs are written to the image table, outputs are read from the image table). Upon completion of I/O updating, normal scanning resumes. If either 8105 or 8106 is *set to "0", only the local digital I/O are* updated (this default does not apply to local register modules).

Bit 8: The Model 650 is capable of complex COMPARE functions (i.e. IF S50 = S60 X S70...). If the integer or floating point math operation within a COMPARE statement results in a value beyond the processor's capability, bit 8 is set to "1". This bit is reset to "0" at the beginning of each new scan.

Bit 9: This bit is set if the Model 650 detects a control register checksum error upon power-up. ERROR 900 or 901 appears in register 8175 unless overwritten by a higher priority error. Also indicates that all control registers have been set to their default values and that all data storage registers have been reset to "0". Typically this bit is set if battery backup has failed for any length of time. Bit 9 is reset by a CLEAR ALL or DATA ENTER operation.

Bit 10: *This bit is set to "1" if an overflow occurs* during a MATRIX or ARRAY operation and is reset at the beginning of a new MATRIX or ARRAY rung.

Bit 11: When this bit is set to "1", the Model 650 interrupts the normal ladder scan and performs up to 5 milliseconds of servicing on any pending communication in the buffer for the PRGMR (Channel 1) port. Normal ladder scan continues upon completion of the communication servicing.

Bit 12: Performs the same function as bit 11, except for the COMM (Channel 2) port.

Bit 13: Performs the same function as bit 11, except for the Ethernet (Channel 3) port.

Bit 14: Not used.

Bit 15: Setting this bit to "1" prevents program editing through either port and any route by any device other than the device which set *the bit to "1". Refer to Section 6.5.*

Bit 16: This "inhibit rung" can be placed within the main ladder program to prevent alteration and/or viewing of certain rungs. Refer to Section 6.7 for more details.

REGISTER 8176 READ-ONLY BITS - The following bits cannot be altered by the user in any way, however, these can be examined as contacts in the program:

Bit 17: This bit is set to "1" when either the Model 650's internal lithium battery or the power supply batteries are low. This bit can be used in conjunction with bit 32 of this register which signifies that only the internal lithium battery is low. Refer to Section 12.1.2 for more information about low battery indication.

Bit 18: "OPEN". This bit is always de-energized, and is used to signify an OPEN in the ladder program.

Bit 19: "SHORT". This bit is always de-energized. A normally-closed contact in the ladder program uses this bit to signify a SHORT.

Bit 20: Reserved for processor internal use.

Bit 21: Set to "1" during the first power-up dummy scan.

Bit 22: Set to "1" during the processor's second power-up dummy scan.

Bit 23: This bit is set to "1" whenever the Model 650 is scanning the ladder program after executing the first two dummy scans.

Bits 24 to 31: Reserved.

Bit 32: Set to "1" when the voltage of the Model 650's internal lithium battery falls below the point needed to maintain memory in RAM. This bit can be used in conjunction with bit 17 to provide a better idea of which battery is defective or missing. Refer to Section 13.1.2 for more information about low battery indication.

8177: Password register. Contains a value that determines if register 8178 controls editing, forcing, and register write restrictions. If a zero is in register 8177, then any value in register 8178 is ignored. When reading register 8177, if a zero is stored then a zero is displayed. If any other value than zero is stored in register 8177, then "-1" is displayed to keep the password a secret. See Section 6.9 for more details.

8178: Restriction register. Each of the first 12 bits in this register determines which security feature is enabled for the PRGMR, COMM, and the ETHERNET port. See the following table to determine which security is in effect when the indicated bits are set to "1".

8178 Bits 1-12:

BIT	SECURITY MEASURE IN EFFECT WHEN BIT IS SET TO "1"
1	Restricts priority writing to registers through the PRGMR port.
2	Restricts the forcing of registers through the PRGMR port.
3	Restricts *non*-priority writing to registers through the PRGMR port.
4	Restricts program editing through the PRGMR port.
5	Restricts priority writing to registers through the COMM port.
6	Restricts the forcing of registers through the COMM port.
7	Restricts *non*-priority writing to registers through the COMM port.
8	Restricts program editing through the COMM port.
9	Restricts priority writing to registers through the ETHERNET port.
10	Restricts the forcing of registers through the ETHERNETport.
11	Restricts *non*-priority writing to registers through the ETHERNET port.
12	Restricts program editing through the ETHERNET port.

NOTE: *The CLEAR ALL operation is always allowed through the PRGMR port.*

GROUP #2: Read-Only Registers

8179: Not used.

8180: Percentage of memory used on high-speed scanning processor (compiled user memory).

8181: Indicates the total amount of ladder program memory (MSW) in bytes available to the user. This figure reflects intermediate memory.

8182: Indicates the amount of ladder memory (LSW) in bytes that is available for use. This figure reflects intermediate memory.

8183: Stores information on some types of errors. See Appendix A.

8184: Same as 8183. See Appendix A.

8185: Indicates the number of rack addressing rungs that are programmed into the Model 650.

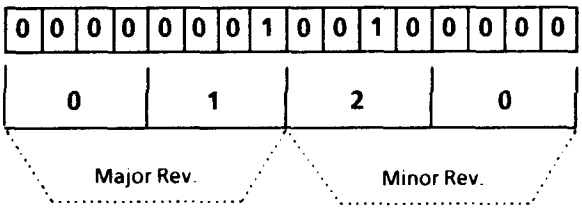
8186: Primary Processor Status Register. The individual bits of this register are used for indicating certain processor conditions. These bits can be examined, but cannot be altered in any way. See the following table for an explanation of each of the bits in register 8186.

8186 Bits 1-16:

BIT	MODEL 650 STATUS WHEN BIT IS SET TO "1"
1	Halted
2	Outputs disabled
3	Running
4	MEMORY LED is ON
5	FORCE LED is ON
6	I/O LED is ON
7	Key is in HALT position
8	Key is in DISABLE OUTPUTS position
9	Key is in RUN position (if bits 7, 8, and 9 are "0", then key is in RUN/PROGRAM position)
10	WRITE PROTECT LED is ON or FLASHING (hardware or software security is enabled)
11	BATTERY LOW LED is ON or FLASHING.
12-16	NOT USED

8187: Indicates the number of ladder rungs that are programmed in user memory.

8188: The three most significant digits of the 16-bit data field of this register identify the *type* of Model 650. The least significant digit identifies the major revision level. (i.e., for the 16K version, rev 1.XX, the digits are 6541; for the 26K version, rev 1.XX, the digits are 6551). The 16-bit Status field stores both the major and minor revisions for the Control Processor software. The data for this field is stored in a Binary Coded Decimal (BCD) format. The first two BCD digits (most significant byte) show the major software revision, followed by an implied decimal point. The last two BCD digits (least significant byte) indicate the minor revision of the software. For example:



Register 8188 Status field

Register 8188's status field indicates a major software revision of 1 and a minor revision of 20. This is read as revision 1.20 of the control processor software.

8189: Memory word information. Bits 1 through 8 indicate total intermediate memory available in bytes (MSW). Bits 9 through 16 indicate the number of bytes that make up a user word.

8190: Total intermediate memory available in bytes; *Least* Significant Word (LSW).

8191: Intermediate memory used in bytes; *Most* Significant Word (MSW).

8192: Intermediate memory used in bytes; *Least* Significant Word (LSW).

14 TECHNICAL DATA

This section provides supplementary technical details on memory use, scan speed, and general processor operation.

14.1 Memory Utilization

The Model 650 contains *two* copies of the user memory: an *intermediate code* memory and a *compiled code* memory. Intermediate code is the "universal language" that all SY/MAX programmers and processors "speak", while compiled code allows for fast program execution.

Since these two memories reside in separate locations, each has its own allocated space. *Either* memory can be the limiting factor for the user program size, depending on which instructions are used. The programming device's status screen displays the amount of *intermediate* memory used, as well as the amount of available memory.

The amount of *compiled* memory used is monitored in register 8180 as a percentage of the total available memory. As the user program is developed, available memory in the compiled area decreases, indicated by register 8180 approaching 100 (%). In the SCP-655, **compiled memory availability is normally the limiting factor in user program size.** Therefore, register 8180 needs to be monitored to determine the amount of user memory actually remaining for ladder programming.

As can be seen from Figure 14.2A and 14.2B, compiled memory use is generally greater than intermediate memory used for a given instruction. Thus compiled memory space will usually be consumed faster than the intermediate memory space. This means that based on experimentation and observation, a typical user program may run out of memory at approximately 26K of intermediate memory because the compiled memory space has been used. Program capacity could actually be more or less than 26K and is dependent upon the instruction set combination. As the size of the program increases, *both* registers 8180 and the "MEMORY USED" display on the status screen of the programming device need to be monitored to determine the remaining memory space.

The following sections explain memory use for relay circuits, general instructions, and IF/LET/MATH instructions.

14.1.1 RELAY CIRCUITS

In general, the amount of memory required for relay circuits can be determined by counting the number of contacts, parallel branches, and coils in each rung. A contact or coil (including its address number), requires one word of intermediate memory (the coil requires three words of compiled memory). Spaces do not require *any* words of memory.

The sample rung in Figure 14.1 requires eleven words of intermediate memory and 13 words of compiled memory (*), as illustrated by the numbers associated with each contact and output:

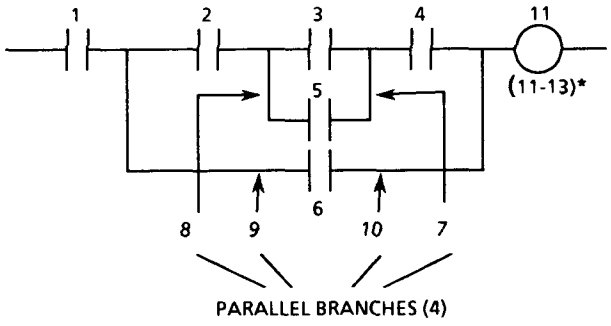


Figure 14.1 Rung Using 11 Words of Intermediate Memory and 13 Words of Compiled Memory

Each rung of intermediate memory also requires an additional 1/3rd of a word. *This 1/3rd word cannot be represented by the "Memory Used" display on the programmer.*

For example, if a contact and a coil are added to user memory, the programmer may indicate that *three* words of intermediate memory were consumed. The third word here is the fractional (1/3rd) word rounded up to a whole word.

14.1.2 GENERAL INSTRUCTIONS

Figure 14.2A shows memory use for most Model 650 instructions. See Section 14.1.3 for special considerations on the IF, LET, and math functions. INT refers to intermediate memory, while COMP refers to compiled memory. Each instruction will consume the indicated number of words from the respective memories.

INSTRUCTION	Words of Memory Used INT	Words of Memory Used COMP
GENERAL RUNGS		
Contact	1	1
Parallel Branch	1	1
Coil	1	3
Coil, Latch/Unlatch	1	2
Coil, Transitional	1	5
Master Control Relay	1	3
Timer	6	7
Counter	6	8
Shift Register (sync. or asynch)	6	5
SUBROUTINES		
MARK/GOTO/Start of Subroutine	4	1
GOSUB	4	3
RETURN	3	1
PRIORITY COMMUNICATIONS		
Register READ/WRITE	6 ▸	4
ALARM Message	5 ▸	4
Print ASCII	2 ▸▸	3 ▸▸▸
Immediate I/O or Communication Update	1	1

▸ Add one word per route.
▸▸ Add one word per register or one word for each two characters.
▸▸▸ Add one word if more than three registers or ALPHA boxes, two words if more than six registers or ALPHA boxes.

Figure 14.2A Memory Use

14.1.3 IF, LET, AND MATH INSTRUCTIONS

In an IF or LET instruction, the amount of memory required is a function of the type(s) of operation being performed. In addition to intermediate memory (INT) used, Figure 14.2B also shows entries for compiled memory (**COMP A** and **COMP B**). From the table in Figure 14.2B, any IF or LET statement (including matrix) requires a fixed quantity of words for the basic instruction, plus one or more additional words for each addition, subtraction, AND, OR, or EXCLUSIVE OR performed in the horizontal box.

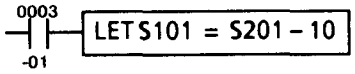
INSTRUCTION	Words of Memory Used INT	Words of Memory Used COMP A	Words of Memory Used COMP B
LET, IF, AND, OR			
LET (data transfer)	3	5	2**
LET (matrix data transfer)	5	5	4**
IF (compare)	3	5	3**
IF (matrix compare)	5	5	4**
AND / OR / EXCLUSIVE OR	1	2	1
Binary-to-BCD, BCD-TO-Binary	1	-	1
INTEGER MATH			
Addition /Subtraction	1	2	1
Multiplication /Division / Square Root	1	-	1
FLOATING POINT MATH			
Addition/Subtraction/Multiplication/Division/Square Root/Sine/Cosine/Base 10 and Base E Logarithms/Y to the X	1*	-	1

* If any number in a LET or IF box is a floating point constant, then two words of memory are required.
** If the first number in the LET or IF box is a floating-point constant, then one extra word is used.

Figure 14.2B Memory Use

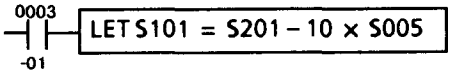
In Figure 14.2B, the **COMP A** column should be used when an IF or LET box contains only integer values and performs ONLY the following operations: compare ($=$, $\neq$, $<$, $\geq$), addition, subtraction, logical AND, OR, or EXCLUSIVE OR.

The following sample rung illustrates this memory use:



This rung requires a total of five words of intermediate memory: one word for the contact, three words for the basic LET itself, and one word for the subtraction operation, as well as eight words of compiled memory (one for the contact, five for the basic LET, and two for the subtraction). *If the above rung was changed to a MATRIX LET, the intermediate memory consumed would increase to seven words, while the compiled memory would remain at eight.*

Use the **COMP B** column in Figure 14.2B whenever a floating-point operation is used within an IF or LET box or if the box includes any operation *other* than compare, addition, subtraction, AND, OR, or EXCLUSIVE OR. If an IF or LET box meets this condition, word usage for the *entire* box should be calculated using the **COMP B** column. The following sample rung illustrates this procedure.



Since this LET box now contains a multiplication, the rung requires a total of six intermediate memory words: (three for the basic LET, one for the contact, one each for the subtraction and multiplication operations, and five words of compiled memory (one for the contact, two for the basic LET, and one each for the subtraction and multiplication)).

14.2 Scanning Techniques

14.2.1 DESCRIPTION

The Model 650 contains a high-speed scan processor, math coprocessor, and control processor. The scan processor executes certain types of tasks in parallel and executes the ladder program faster than the control processor for better throughput. The Model 650 control program execution time is based primarily on 3 factors:

1. Scan time of the user ladder program itself.
2. External I/O Update time (including forcing, if active).
3. Servicing of external communications.

The Model 650 makes use of an Image Memory to hold the state of the first 8000 registers to make logical program decisions. The contents of the Image Memory are exchanged with the external I/O. The present I/O state is transferred to the Image Memory at the end of every scan.

The time necessary to perform this external I/O update is a function of the number of I/O registers assigned to other modules in the system and how many of these registers are fragmented (containing both input and output data in the same register).

Servicing of external communications is normally limited to 5 msec. per scan unless the user sets bit 8170-16 or requests an immediate communications update; refer to Section 14.2.2.

14.2.2 SCANNING SPEED

In addition to the time needed to execute the ladder program, the *scan time* of the Model 650 is composed of the time needed to update the external I/O *plus* the time to service the communication ports. In general, the following equation can be used to estimate scan speed:

$$ST = [K \times (LUP)] + [0.04 \times (ER)] + [0.45 \times (FER)] + COM$$

In the above equation:

ST is the SCAN TIME in milliseconds.

K is the LADDER SCAN SPEED per K of user program (estimated from execution times in Figure 14.3 on the next page).

LUP is the LENGTH of USER PROGRAM in thousands of words (k words).

ER is the number of EXTERNAL REGISTERS.

FER is the number of FRAGMENTED EXTERNAL REGISTERS

COM is the time required to service the three communication ports.

Figure 14.3 provides a list of typical operations and an approximation of associated execution times. For operations not listed, execution times are difficult to estimate because of the complexity of certain operations. Thus, Figure 14.3 provides the user with relative comparison times, indicating which functions are more processor-intensive.

The user is advised that scan times may not be consistent, depending on rung true/false conditions and the amount of communication traffic from Figure 14.3, numerous instructions are listed which are not executed when their input logic is false. For example, executing a repeat MATRIX LET of 100 registers will increase scan time by almost 10 msec when true. The PID instruction is also shown to be scan-intensive. Scan time is also clearly affected by non-sequential program control instructions such as GOTO, GOSUB, and TIMED INTERRUPT. For scan-critical applications, user should be aware of the scan impact of worst-case logical execution paths. Register 8166 (minimum scan time register) can be used to "smooth out" widely varying scan times.

Refer to Section 14.2.3 for information on optimizing scan speed.

CAUTION

When editing while the processor is in RUN/PROGRAM, the program scan time may increase significantly as discussed below.

When performing an edit while the processor is executing the user program, the normal processor scan time may be momentarily increased. The increased scan time is a one-time occurrence and is caused by the processor loading the new rung(s) into the location in memory where the program resides.

Depending upon the program complexity, the type of edit performed, and the memory location where the edit takes place, the scan-time may increase several hundred milliseconds. The longer and more complex the program (i.e., multiple MARK rungs, GOTO, and SUBROUTINE commands) and the earlier in memory that the edit takes place, the greater the scan-time increase.

For any given program, the worst-case scan time can be measured by monitoring register 8172 (scan-time register) while performing an insert of rung 1. If an excessive increase in scan time cannot be tolerated while scanning the user program, edits can be prevented by leaving the keyswitch in RUN. Alternately, register 8167 (programmable scan limit) may be preset to halt program execution based on a user-specified maximum scan time limit.

A processor which is called upon to service its communications ports (such as when a programmer is connected) will also experience a worst-case scan time extension up to 5 msec at End of Scan (EOS). Note this is 5 msec *total* for all 3 ports. An Immediate Comms Update instruction also exists (bit 11, 12, or 13 of register 8176) which allows the user to interrupt scanning to service the communication ports more frequently, allowing better Comms throughput at the expense of longer overall scan time.

Also, a special control bit (8170-16) may be set by the user to cause the processor to *completely* finish servicing an entire message block at the end of scan. This ensures a given message block will have all registers incorporated into the image table at the end of the same scan. The potential scan time impact is to add on up to an extra 5 msec (10 msec total) at EOS. Refer to Sections 5.6.1 and 13.2 for more about Register 8170-16.

INSTRUCTION TO BE EXECUTED	EXECUTION TIME IN MICROSECONDS (Except where noted)
	ALL MODEL 650s
Contact*	0.8
Coil (Including MCR and latch/unlatch)	1.4
Transitional Coil	1.4
Timer (Disabled/Enabled)	14.4/18.2
Counter (Disabled/Enabled)	20.5/23.0
IF* (Integer/ Floating-point)	12.0/122
LET* (Integer/ Floating-point)	10.0/128
Matrix LET* (Integer)	97 per register
Math (Integer/Floating-point):	
ADDITION	5.3/46.0
SUBTRACTION	5.3/46.0
MULTIPLICATION	122/46.4
DIVISION	122/46.4
Logical AND, OR, XOR	3.7
Shift	96 + 2 per register

Figure 14.3 Execution Times For Typical Operations

INSTRUCTION TO BE EXECUTED	EXECUTION TIME IN MICROSECONDS (Except where noted)
	ALL MODEL 650s
FIFO	102 + 2 per register
PID* (Rev. or Dir.)	740 (typical)
PID* (Manual)	116 (typical)
Immediate I/O Update†	140 + 40 per register
T WRITE, T READ, T ALARM	133
T PRINT	183 (typical)
SIN/COS/TAN	239
Ln/LOG	154
Y ** X	352
Add Stat	335
Variance	273
T WRITE, T READ, T ALARM	133
TPRINT	183 (typical)
SIN/COS/TAN	239
Ln/LOG	154
Y** X	352
Add Stat	335
Variance	273
MARK ST SUB	0
MARK	0.5
GOTO*	1.4
GOSUB* (with RTN)	72.0
External I/O Update Time†	320 per 8 local unfragmented registers per scan
MIX #1**	0.8 milliseconds per K of ladder
MIX #2***	3.2 milliseconds per K of ladder
MIX #3****	2.7 milliseconds per K of ladder

Figure 14.3 Execution Times (continued)

FIGURE 14.3 CODES

† *123 microseconds per register for Class 8030 Type RIM121 analog input, 75 microseconds per register for Class 8030 Type ROM121 analog output.*

* *If rung is TRUE up to that point; if FALSE, the instruction is not executed and time is reduced. See Section 14.2.3, "Rung Execution Algorithm Guidelines".*

** *Mix #1 is composed of all Boolean rungs.*

*** *Mix #2 is 50% Boolean, 34% IF and LET rungs, 8% counters, and 8% timers.*

**** *Mix #3 is 80% Boolean, 5% timers, 5% counters, and 10% addition/subtraction.*

CONSIDERATIONS FOR "TIME MIXES" #1-#3

- On an average, only 66% of any rung must be executed.
- Boolean assumes an average of two contacts for each coil.
- External I/O Update or communication processing time is not included in any of the mix execution times.

14.2.3 OPTIMIZING SCAN SPEED

In some ways, optimizing the scan speed of a Model 650 processor is no different than the procedure used for any computing device. For example, the Model 650 program is compiled from intermediate code into a 64-bit word compiled format that allows rapid execution by the scanning processor.

While this intermediate to 64-bit compiled code conversion operation is transparent to the user, the Model 650's instruction set contains scan control instructions like GOTO and GOSUB that allow skipping over sections of a program that do not require scanning at a particular time. This "skipping over" process saves scan time and improves performance (throughput). This simple technique should not be overlooked when dealing with programs containing sections that do not require constant scanning.

When laying out and programming any system, applying knowledge of how that system behaves to the system's configuration can optimize its performance.

Over the next few pages are some simple guidelines (most of which have been previously discussed in this manual) for economizing scan time.

RACK ADDRESSING GUIDELINES

Minimizing external I/O update time using good rack addressing techniques is demonstrated by the example in Section 3.6.5. When Rack Addressing a system containing Local Interface (LI) module(s), scan speed is optimized by assigning *only those registers necessary for remote I/O operation* to an LI. Any registers (up to 255 - recall that registers above 255 aren't transferred) unnecessarily assigned to an individual LI results in needless end-of-scan register transfers between the Model 650 and the LI's data storage registers.

The equation of Section 14.2.2 shows that updating an unfragmented register on the bus requires 40 microseconds. If the conditions of the example in Section 3.6.5 are used, the unnecessary assignment of all 512 LI module registers decreases throughput: *Update Time for 512 registers = 255 x 0.04 ms =* **10.2 ms per scan**, *and the Update Time for 50 registers = 50 x 0.04 ms =* **2.0 ms per scan**

SUMMARY:In the scan time calculations, 8.2 milliseconds is needlessly added to the scan time for unnecessarily updating over the bus. Thus, assign only needed registers for external I/O to the LI module(s).

When assigning registers to other register modules in the CPU rack, ONLY assign as many registers as needed by the module. For example, assign only one register to a 16-point digital I/O module, or four registers to a four-channel analog module. Any registers needlessly assigned to the module results in the 40 microsecond per register scan penalty.

EXTERNAL I/O REGISTER GUIDELINES

The equation of Section 14.2.2 also shows that the time required to update a register increases tenfold when the register is *fragmented* instead of unfragmented. A fragmented register refers to an external I/O register that contain both input and output points. Thus, when using four-point and eight-point digital I/O modules, lay out the programmable control system so that *all* points of a particular external I/O register are assigned as *either* inputs *or* outputs.

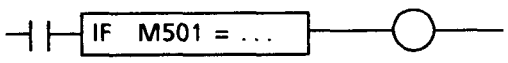
For example, if a Model 650 is being used in an HRK-type rack (eight-function rack), both modules associated with an external I/O register should be either inputs or outputs. This principle applies to remote I/O registers as well.

NOTE: *Fragmenting registers forces the processor to make decisions that consume valuable processing time. These time-consuming decisions can be eliminated if external I/O registers are made up strictly of inputs or outputs.*

RUNG EXECUTION ALGORITHM GUIDELINES

The Model 650 executes ladder rungs in such a way that elements of conditional logic may actually be skipped over if the logical outcome of the rung is no longer in doubt. This applies to logically TRUE as well as logically FALSE rungs.

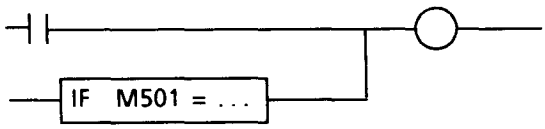
A good example of this is shown in the following:



The matrix IF instruction shown above only executes if the leading contact is CLOSED. If OPEN, the matrix IF is skipped over (no pointer information is updated) and the coil is turned OFF. If the leading contact is CLOSED, the matrix IF is scanned to determine the state of the coil. If the order of the conditional elements was inverted (i.e., the matrix IF came *before* the contact), although logically equivalent, the execution time could increase dramatically for the FALSE condition since in this configuration the matrix IF would *always* be executed.

If proven FALSE, the contact is skipped over; the savings in scan time, however, would be negligible compared to the savings if the matrix IF were not scanned at all.

A similar principle applies to saving processor time for the TRUE condition shown below:



If the contact shown above is closed, the coil is ON regardless of the condition of the matrix IF instruction. In this case, the matrix IF doesn't need to be executed and some scan time could be saved in the process.

However, if the matrix IF and the contact were interchanged, the matrix IF always executes first. This may have an adverse effect on the rung's execution time.

Another simple rung execution technique can be applied to LET statements. Since the LET only executes if the rung is TRUE, any initialization constants scanned more than once will needlessly consume processing time and should be avoided in the user program.

SUMMARY: When placing a premium on optimizing scan time, examine rungs for logical equivalence to see if interchanging conditional elements will reduce the execution time.

SCANNING PROCESSOR VERSUS CONTROL PROCESSOR GUIDELINES

Any functions that cannot be handled by the scanning processor must be "handed off" to the control processor. Since the control processor is also tasked with handling other system interrupts, longer scan times can result.

Such multitasking operation makes it very difficult to quantify precise execution times for these instructions since it cannot be predicted in advance how many other demands (such as communication handling) are being made on the control processor at the time of the "hand-off".

As shown in the table of Figure 14.3, execution times can vary widely. This variation occurs because some of the instructions are executed by the scan processor while others must be handled by the control processor. For example, integer addition and subtraction are handled by the scan processor, while integer multiplication division and all floating-point operations are handled by the math coprocessor.

Since integer multiplication and division and floating-point math operations are handled by the *math* coprocessor, the *control* processor must act as an intermediary between the scanning processor and the math coprocessor. While this is occurring, the scanning processor remains idle until the result is computed. This makes for an efficient use of processing power.

The following provide additional considerations when comparing the operation of the scanning processor with the control processor:

- Although GOSUB and RETURN instructions have to be executed by the *control* processor, their careful use can result in a great savings in throughput.

- Shift register and FIFO operations are executed by the *control* processor; their execution times are affected by the number of registers involved.

- Matrix operations must be performed by the *control* processor and can significantly influence scan time depending on their size and pointer configuration. Each matrix operation requires approximately 97 microseconds to calculate a new element.

  - When using *matrix LET* statements, use "level" or "incremental" LETs over "repeat" LETs wherever possible. This distributes the increased processing time over many scans.

  - For *matrix IFs*, use "level" instead of "repeat" whenever possible. If "repeat" *must* be used, follow the principles given in the previous *Rung Execution Algorithm Guidelines* section to save scan time.

14.3 Theory Of Operation

The Model 650 consists of several major subsystems, all under the control of a central processing unit. Refer to the block diagram in Figure 14.4 while reading this section.

The control processor is a Motorola Type 68010 and is responsible for performing and/or coordinating all functions of the Model 650, including interrupts and error conditions from the serial communications interface, SY/MAX bus, scan processor, and math co-processor.

Scanning in the Model 650 Processors is performed by an Advanced Micro Devices Type 29116 processor utilizing a 64-bit compiled word.

Floating-point operations are performed by a Motorola Type 68881 math coprocessor. The system clock runs at 8 MHz.

Each subsystem within the Model 650 is briefly explained in the following.

Control Processor: This device either performs or coordinates all of the processor's functions, including communication via the two serial RS-422 ports. The control processor also compiles the ladder program in the Compiled User Memory, controls interrupts and error conditions, and handles SY/MAX bus duties.

Math Coprocessor: Allows the Model 650 processor to perform floating-point math operations.

Scan Processor: Performs computation of the Model 650's output states based on the current condition of inputs and registers. The computations and sequence in which they are performed are determined by the Compiled User Memory. The I/O states and register values are obtained from the Image Memory.

Image Memory: The Image Memory consists of battery-backed RAM that provides 16-bit data and status fields for user registers 1 through 8000 and control registers 8093 through 8192. Parity protection is provided for this memory. This memory contains both the internal and external I/O and registers.

Between scans, the control processor outputs the current values in this memory to the appropriate external devices. The control processor also updates the Image Memory according to the external I/O registers to prepare for the next scan.

User Memory: Partitioned into two distinct portions for maximum efficiency in scanning and program manipulation:

1. *Intermediate Code User Memory*: This battery-backed memory occupies 32K words of the control processor's operating (scratch) memory. The Intermediate User Memory is only accessible by the control processor. It serves as the source for the compile operations which generate the instructions for the Compiled User Memory.

2. *Compiled Code User Memory:* Consists of battery-backed CMOS static RAM that provides 32K of 64-bit word user program storage. Each word is composed of the scan processor instructions and control codes. This memory serves as the executive memory for the scan processor and can be randomly accessed by the control processor for program loading and editing purposes.

Serial Communication Interface: Controls the two RS-422 ports on the front of the Model 650 (PRGMR/Channel 1 and COMM/Channel 2). More information on these ports is found in Section 5.

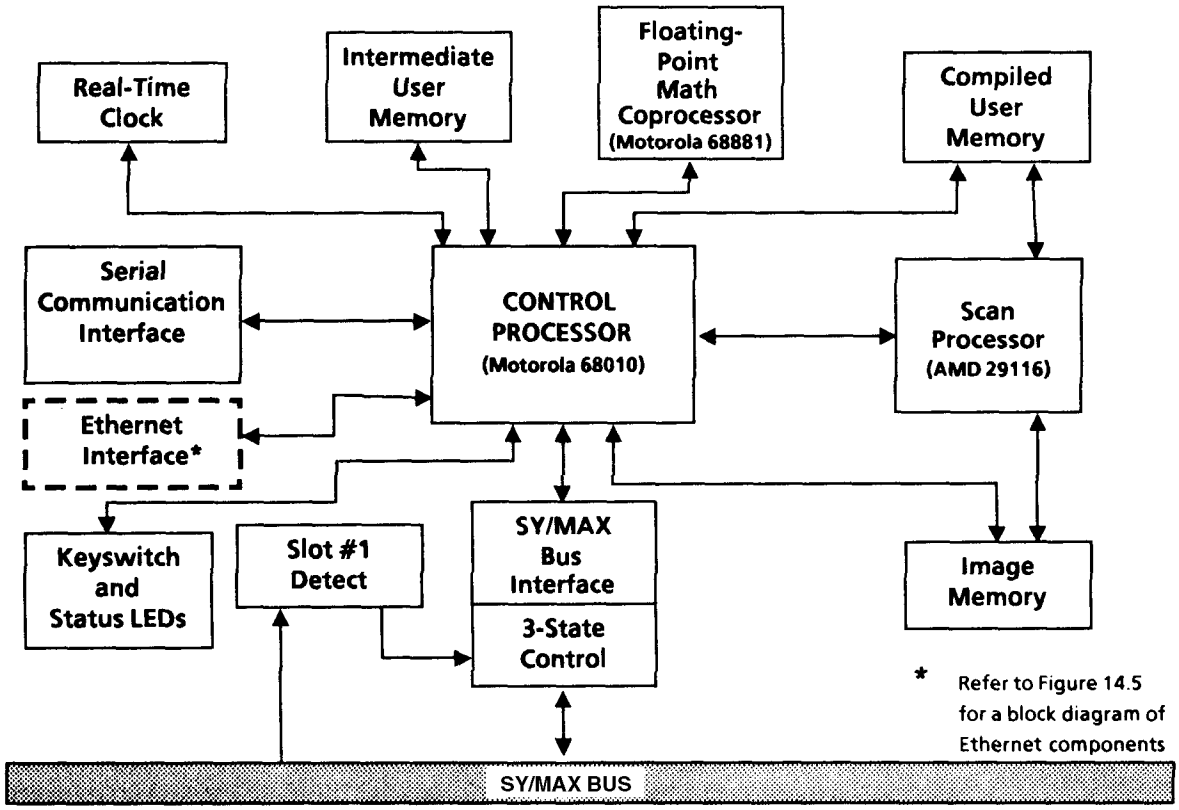


Figure 14.4 Model 650 Block Diagram

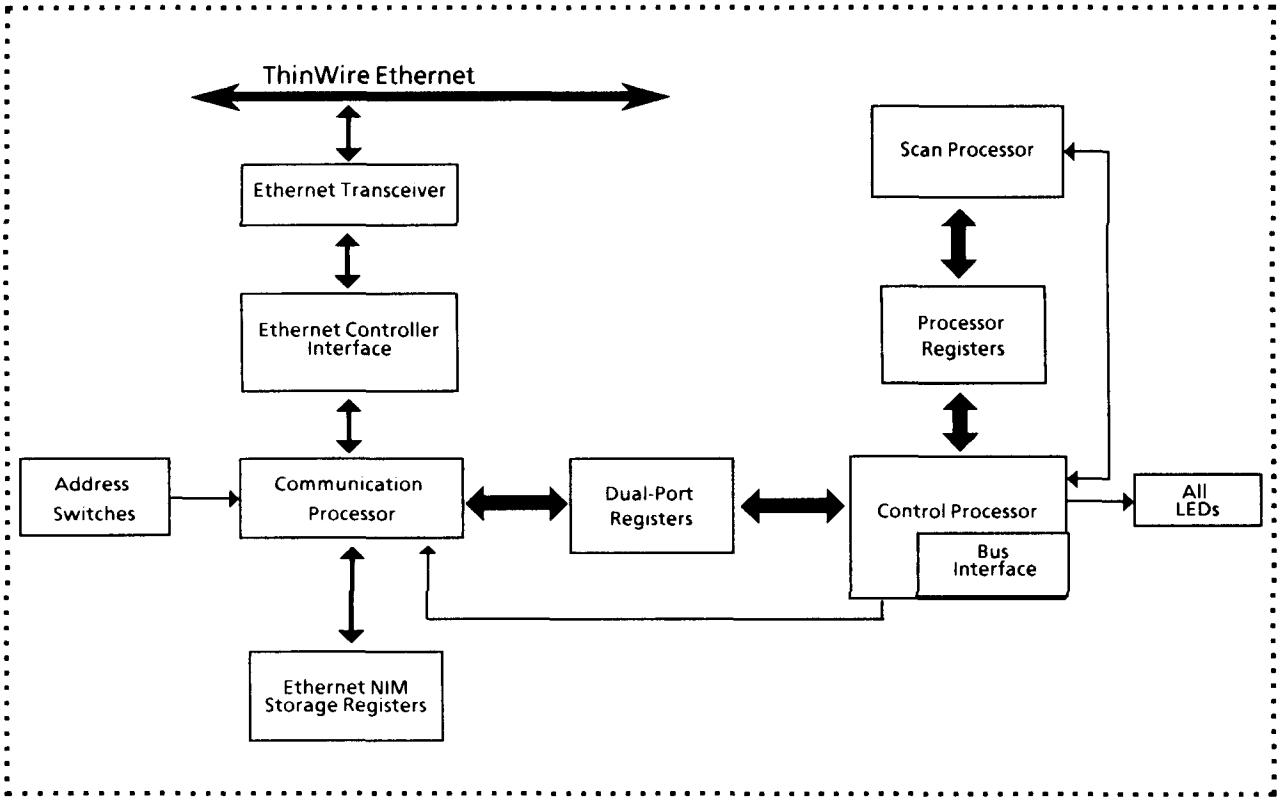


Figure 14.5 Model 650 Ethernet Interface

SY/MAX Bus Interface: Allows the Model 650 to control a SY/MAX digital or register rack as a bus master.

Ethernet Interface: Controls the Ethernet port (Channel 3). The Ethernet Interface converts data from the Ethernet packet into SY/MAX format so it can be processed by the control processor. Refer to Figure 14.5 above.

Real-Time Clock/Calendar: Provides the Model 650 with time-of-day and date information. This information can be used throughout the entire programmable controller system. For more information on the clock see Section 12.2.

14.4 Forcing

WARNING

Since forcing overrides the actual logic of the control program, ONLY QUALIFIED PROGRAMMERS who understand the contents of this section should use the forcing capability. Mistakes in forcing may cause equipment damage or bodily injury.

Forcing allows an external input or output to be turned ON or OFF via the programmer's keyboard and overrides the ladder logic state of the external field device. Forcing is permitted from all ports.

In the Model 650, forcing is implemented just prior to updating the external I/O at the end of scan. *The forced state of an output WILL NOT be reflected in the image table, meaning that the user program logic executes and is based on the actual image table logic state and not on the forced state.* This operation differs in comparison to other SY/MAX processors.

The forced state of an input IS written to the image table. Since forcing is only implemented in the external I/O at the end of scan, forcing an internal relay or data storage register does not alter the actual logic state in the user ladder program. The programmer indicates the bits of an internal relay or data register to be forced even though they are *not* actively changed. This operation differs from other SY/MAX processors and is discussed in the following.

Since forcing is implemented on external I/O and *not* in the image table, the following rules govern *what is observed* and *how the programmable controller system behaves.*

FORCING AN EXTERNAL OUTPUT

A forced external output causes the associated external field device to assume the forced state. The programming device reflects this when displaying a forced rung by displaying an "F" within the forced element. If relay contacts with the output address are used in the program, they will assume the logic state of the output as executed in the image table. Since the relay contacts and coil displayed on the programmer reflect the forced state and *not* the image table logic state, this may conflict with the way the logic is actually being executed.

EXAMPLE: Consider the rungs shown in Figure 14.6. Contact 13-02 is an external input, while coils 14-16, 13-11, and 13-12 are external outputs. The following conditions apply to the rungs:

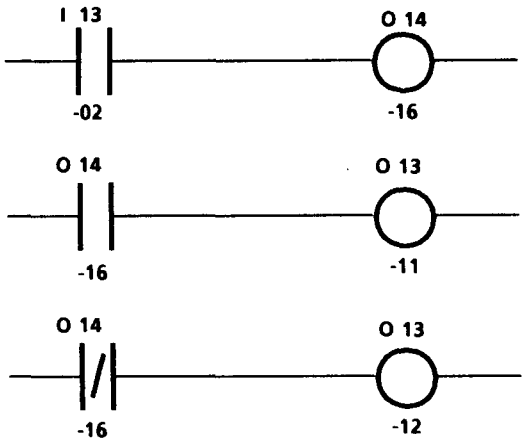


Figure 14.6 Forcing Example

CONDITION: Output **14-16** is forced ON, while input **13-02** is OPEN.

RESULT: The external device wired to output **14-16** is energized, and the programmer inserts an "F" inside the coil and relay contacts with address **14-16.** The programming device displays the output contacts and coil forced ON to reflect the forced state.

COMMENTS: Even though the N.O. relay contact **14-16** is forced "ON", output **13-11** is OFF while output **13-12** is ON. This occurs because coil **14-16** and its associated relay contacts are actually OFF in the image table (because contact **13-02** is OPEN).

If contact **13-02** is CLOSED instead of OPEN, the image table logic coincidentally agrees with the forced state; output **13-11** is ON while output **13-12** is OFF.

If it is desired to have the contacts of coil **14-16** follow the logic state of the coil, the proper technique is to force external input **13-02** as discussed in the following:

> **WARNING**
>
> **Forcing an unlatch coil "ON" actually latches the coil. Beware of the potential damage to equipment or bodily injury.**

FORCING AN EXTERNAL INPUT

A forced external input overrides the actual condition of the input field device and causes the associated bit address in the Model 650's image table to assume the forced state. The programming device indicates the forced state. As long as the forced external input bit is not altered in the ladder program (can only happen if the input bit was incorrectly programmed as a coil), this forced bit state is used when the ladder program is executed.

EXAMPLE: Assume the same rungs are used in this example as in the previous "EXTERNAL OUTPUT" example.

CONDITION: Input **13-02** forced ON, external field input is OFF.

RESULT: As long as the state of bit **13-02** is not altered in the user ladder program, the forced state is preserved in the image table no matter what the state of the external field device's input is. This means that output **14-16** is actually turned ON *in the image table* along with the external device connected to output **14-16.** The relay contacts of coil **14-16** properly reflect the state of the coil. Output **13-11** is ON, while output **13-12** is OFF.

FORCING AN INTERNAL RELAY EQUIVALENT OR DATA STORAGE REGISTER

As discussed previously, forcing is a condition which applies only to *external* I/O.

NOTE: *Attempting to FORCE an internal relay equivalent or a bit in a data storage register has NO EFFECT on its state in the image table when used to execute the ladder program. This is true even though the programming device displays the FORCED state.*

Since there is no external I/O associated with a forced internal relay equivalent or storage register bit, the Model 650 has no way of implementing the forced state. However, the programming device shows the bit as forced; *this may conflict with the way the rung is actually being executed.*

This situation is somewhat analogous to the behavior of contacts that use the same address as a forced coil. The contacts *indicate* a forced state, but in actuality assume the logic state of the output as executed in the image table. In the case of an internal relay there is no external output associated with the address. The Model 650 has nothing to force externally, and the internal relay contacts continue to follow the logic of the coil as instructed by the user program.

To summarize, perform forcing *ONLY on external inputs and outputs*. Forcing an *internal* relay equivalent or bit in a data storage register has no effect on the register and can produce information that is subject to misinterpretation on the programming device screen.

APPLICATION CONSIDERATIONS FOR FORCING

- Any external I/O register is forcible. This includes local digital I/O in a register rack and digital I/O controlled via the Parallel Digital Driver/Receiver (PDD/PDR) modules.
- Registers above 8000 are not forcible.
- Bits in a maximum of 256 registers can be forced simultaneously.

- Forcing can *always* be cleared by cycling power to the Model 650.

- Forcing can be inhibited or if already in effect, prevented from being altered by placing the keyswitch in the RUN (as opposed to RUN/PROGRAM) position. See the note following these application considerations.

- When the TIMED INTERRUPT subroutine is programmed and the Model 650 is in the RUN/PROGRAM mode, forced I/O *can* be released by the *FORCE - CLEAR ALL* command.

- Software security can be invoked to restrict forcing on a port basis; refer to Sections 6.9 and 6.11.

- Each forced register adds approximately 100 microseconds to the I/O update time at the end of the processor's scan, regardless of the number of bits forced in the register. Thus, forcing *one* bit in a register causes the same time penalty as forcing all 16 bits in that register.

NOTE: 3 *Special Cases of Forcing Exist:*

Case 1 *If a force command is issued through port 3 only, and both ports 1 and 2 are disconnected or powered down, forcing is not released.*

Case 2 *If a force command is issued through port 1 or port 2 and both ports 1 and 2 are disconnected or powered down, forcing is released.*

Case 3 *If a force command is issued through port 1 or port 2 and port 3 and both ports 1 and 2 are disconnected or powered down, forcing is released.*

14.5 Power Up/Down Sequence

PROCESSOR POWER-UP SEQUENCE

Upon application of power to the Model 650, the following start-up sequence occurs:

1. External outputs are turned OFF.

2. A self-test sequence is performed to determine proper operation. If any part of the self-test sequence fails, the Model 650 will not go into RUN.

The following components are checked when the self-tests are run:

- The executive PROM is checksummed and validated
- *User memory*
- Intermediate User Memory
- Internal RAM
- Compiled User Memory
- Image Memory
- Scan Processor operation
- Control Processor operation
- Watchdog Timer
- *Math Coprocessor*

3. Memory Corruption Test. If an error is found, the MEMORY LED is turned ON and the Model 650 does not execute the existing programmed memory.

4. A minimum 1-second delay is generated in order to allow I/O data to stabilize. The delay may be longer depending on the register cards installed in the processor system and the quantity of memory in the processor.

5. The I/O forcing memory is reset inside the Model 650, thereby releasing all forced I/O.

6. Input status is updated in the Image Memory.

NOTE: *The total time required for all power-up sequence activities is about twelve seconds. During this time, the Model 650 does not respond to any communication activity, and any attached programming device appears to be inactive (in reality, the programming device is being requested to "wait" by the Model 650).*

7. If the processor keyswitch is in the RUN, RUN/PROGRAM, or DISABLE OUTPUTS position, the sequence continues with the "Processor Scan Start-up Sequence" described in the following.

PROCESSOR SCAN START-UP SEQUENCE

Upon changing the processor keyswitch from any position to either the DISABLE OUTPUTS, RUN, or RUN/PROGRAM position, the following sequence occurs:

1. A self-test is run which checks that:
   - A valid user program is loaded
   - All user-programmed MARKs and related GOTO and GOSUB rungs are properly programmed.
   - All programmed I/O devices are checked
2. Two dummy scans are conducted:
   - *Dummy Scan #1* simulates the presence of an MCR in the OFF state and executes the scan so that DISABLE OUTPUTS is ON. Subroutines are scanned first, then the regular ladder program is scanned. This scan has the effect of turning OFF all coils, except for those programmed as internal latches. Outputs in the image table are not transferred to the I/O devices during this scan. The timed interrupt is not executed. During this scan, bit 21 of register 8176 is set to "1".
   - *Dummy Scan #2* removes the simulated MCR, but keeps DISABLE OUTPUTS set ON. The ladder is scanned normally. During this scan, timed interrupts are not executed. At the end of this scan the image table is transferred to the external I/O and programmable interrupt scheduling begins. During this scan, bit 22 of register 8176 is set to "1".
3. Normal ladder program scanning begins. Bit 23 of register 8176 is set to "1".

NOTE: *The total time for keyswitched start-up activities is about five seconds. During this time the Model 650 does not respond to any communication activity. An attached programming device appears to be inactive (although in reality its simply being asked to "wait" by the Model 650). If the programmable control system includes a serial interface module (LI, LTI, or LAI), start-up time may be up to 15 seconds if the serial interface is showing an error condition.*

PROCESSOR POWER-DOWN SEQUENCE

When the Model 650 processor enters a halted state or if power is removed from the system, the Model 650 performs an orderly shutdown procedure that resets all external outputs to their OFF state and halts the memory scan. If Local Interface (LI) modules are used in the system, outputs can be specified to FREEZE instead of turning OFF; refer to the Local Interface Instruction Bulletin #30598-247-XX.

The programmable controller system power supply continues operating, with no interruptions in the operation, through any power loss or "brown-out" condition of approximately 16 milliseconds or less. After this period of time, the processor shuts down and restarts when proper power is resumed.

In addition to all external outputs being reset to OFF in the modules, the *registers* that are rack addressed to the output modules are also reset to 0 in the processor image table. This occurs not only during the CPU power-down sequence, but also during run-to-halt transitions.

14.6 Run-time Operational Checks

The Model 650 performs a number of internal diagnostic checks. Among these are:

1. A continuous check made to ensure that the internal clock driving the processor logic is fully operational (hardware watchdog timeout).

2. Hardware circuitry halts the processor in the event the microprocessor fails to report every 2.5 milliseconds (software watchdog timeout).

3. Each word of user ladder and register memory incorporates a parity bit. The processor checks each examined word as it is executed for proper parity. If an error is detected, the processor halts operation and turns external outputs off. (In this case, the HALT LED flashes and the MEMORY LED is illuminated.)

4. The control processor gives the scan processor a rung to execute at the end of each scan (a "fault rung"). If the rung is executed incorrectly, all outputs are turned OFF and an error is generated.

15 ETHERNET COMMUNICATIONS

15.0 A Glossary Of Terms

The following terms are commonly used throughout this manual and the SFW390 (30598-737-XX) *Instruction Bulletin. They apply primarily to the* Model 650 and SFW390's version of Ethernet:

ACK - A response from an external device to the sender of a data message. An ACK indicates that one or more messages were received and accepted by the destination.

Alarm Queues - Areas of memory supported by the SFW390 software and designed to hold three different classifications of alarms: Faults, Alerts and Warnings. Faults are classified as the most severe and Warnings are classified as the least severe.

Application - A specific process or task, such as tracking and comparing production per machining device, to which a computer solution can be applied.

Asynchronous - A mode of processing in which a user task can continue to execute after a read or write without having to wait for the request to be fulfilled. (See synchronous.) This would allow multiple reads and/or writes to be outstanding at the same time, for example. Register reads and writes using SFW390 or Model 650's can be either synchronous or asynchronous.

Bridge - Similar to a repeater in its ability to connnect segments of a network. It also has the capability to screen information being transferred between segments of a network.

CSMA/CD - Carrier Sense Multiple Access with *Collision Detection. This is a method for controlling* access by devices all sharing a single transmission medium. Other methods include token passing and token ring.

Ethernet or Equivalent NIM - The hardware and firmware internal to the Model 650 programmable *controller that serves as an embedded Network* Interface Module and functions to:

- Send/receive Ethernet packets to/from the VAX computers and other Model 650 programmable controllers.
- Store/retrieve SY/MAX data to/from the Ethernet (Equivalent) NIM registers;
- Send/receive SY/MAX packets to/from the Model 650 programmable controller.
- There is also an equivalent NIM component of the SFW390 software that functions in the same manner as the Model 650's equivalent NIM. The equivalent NIM portion of the SFW390 is often referenced in the SFW390 manual as the SFW390 Ethernet Configuration Registers. Figure E.2 in Appendix E illustrates the location of the equivalent NIM in both the programmable controller and SFW390.

Ethernet - A specification for local communication networks that employs coaxial cable as a passive communications medium to interconnect different kinds of computers, information processing products, and office equipment at a local site.

External Device - Refers to a Model 650 programmable controller, or another computer running SFW390 software (or equivalent) connected to the host computer via the Ethernet network.

Gateway - In LANs, a computer system and its associated software that permit 2 networks using different protocols to communicate with each other. A gateway translates all protocol levels from physical layer up through applications layer, and can thus be used to interconnect networks that differ in every detail.

Global Alarm Queues - System-wide alarm queues, accessible from any application program. SFW390 supports global alarm queues which are specified in the configuration file. Any external device or internal process can write to a global queue and any application program can view, clear or acknowledge entries from them.

Global Mailbox Registers - System-wide registers, accessible from any application program. Global mailbox parameters are specified in the configuration file. Any external device or application program can read or write the global mailbox registers.

Host Computer - A computer attached to a network that provides such services as computation, database access, or special processes or programming languages. The VAX computer running with VMS operating system is the host computer when using SFW390.

IEEE 802.3 - IEEE 802.3 is a specification for a communications standard written by the Institute of Electrical and Electronic Engineers based on the Ethernet specification. Sometimes referred to as ThickWire Ethernet (10BASE5)

IEEE 802.3a - This is a sub-specification of 802.3, which is based on the ThinWire Ethernet specification (10BASE2).

LAN - Local Area Network. A cabling system, usually installed at the periphery of a building, which uses a proprietary protocol and enables a single vendor's equipment to transmit or receive data.

Local Alarm Queues - Alarm queues belonging to and managed by an individual application program. The SFW390 supports both local mailboxes and local alarm queues. Any external device or internal process can write to a task's local queue but only the owner task can view, clear or acknowledge entries from them.

Local Mailbox Registers - Mailbox registers belonging to and managed by an individual application program.

Mailbox Registers - Areas of memory used to emulate a set of SY/MAX general purpose registers. Also called a "register file." Each application program may have a set of local mailbox registers. In addition, there may be a set of global mailbox registers. The principal purpose of these mailbox registers is to allow application programs to receive unsolicited messages.

NAK - A response from an external device to the sender of a data message. A NAK may cause several messages to be retransmitted.

Network - A group of computing devices (computers, programmable controllers, etc.) that are connected to each other by communication lines to share information and resources.

Node - An end point of any branch of a network, or a junction common to two or more branches of a network.

Protocol - A set of mutually agreed rules for exchanging information between computers and computer-based equipment.

Reference - A calling convention in which a variable's address is passed to the function, that is, by reference.

Repeater - A device used to extend transmission ranges/distances by restoring signals to their original size or shape.

Reply Timeout - When a reply to a command originated by a user process is not received within the reply timeout period, the function generating the command will set the status variable appropriately and (for synchronous operation) return an error in the function return.

Response Timeout - When a response (an ACK or NAK) from a destination device is not received within the timeout period, the message will be retried. This timeout applies to a response (i.e. ACK or NAK) from a transport protocol entity on an external device, and should not be confused with a reply timeout.

The response timeout period is user-adjustable. Refer to Section 15.6.2 for more information.

Retry - After a NAK or a response timeout a message will be retransmitted. The number of retries that will be attempted for a message is user-adjustable.

Segment - A continuous length of cable used to connect devices. ThinWire Ethernet defines a segment as being RG-58 cable and extending no more than 185m.

Synchronous - A mode of processing in which a user task issues a read or write request and then waits until that request is fulfilled before continuing to execute. (See asynchronous.) All register reads and writes using SFW390 can be either synchronous or asynchronous.

Terminator - A special connector used on both ends of an Ethernet segment. This connector provides the 50-ohm (or other value) termination resistance needed for the cable.

ThickWire (10BASE5) - Standard Ethernet baseband coaxial cable that serves as the backbone transmission medium for the Local Area Network. Primarily used for facility-wide installations.

ThinWire (10BASE2) - A wiring scheme which uses a type of (thin) coaxial cable for use in Ethernet. Primarily used in office environments. A DEC trademark used to describe its IEEE 802.3 compliant Ethernet products used for local distribution of data communications.

15.1 Introduction

Ethernet is a data communications network used to allow computers and electronic devices to 'talk' to each other. This is accomplished with a coaxial cable and the appropriate Ethernet software on each device connected to the cable (and hence the network). Ethernet is a non-deterministic network, which essentially means that it can not be determined when a particular device will have an opportunity to 'talk' on the network to another device. Ethernet operates on a first-come, first-serve basis. Other devices must wait until the first device is done communicating, and then attempt to be the first one to talk again. (This method is known as CSMA/CD or Carrier Sense, Multiple Access/ Collision Detect. CSMA/CD is a data communications networking standard classified in the IEEE 802.3 standard.) Since Ethernet is non-deterministic, there must be some method of waiting and attempting to access the network if it is currently serving some other device. Thus the concept of timeouts and retries are utilized to provide for multiple attempts to 'talk' on Ethernet. The SY/MAX Model 650 provides a way for Programmable Controllers to 'talk' on Ethernet by using standard ladder programming communication rungs (READ, WRITE and ALARM) for communication port 3. See Section 5.6 for detailed information on SY/MAX communications.

This section is intended as a more detailed analysis of the inherently non-deterministic nature of Ethernet communications. The SY/MAX Model 650 is designed to get up and running with minimal configuration. Upon setting up the Model 650 for the first time, a user simply needs to select a unique SY/MAX drop number for Ethernet and follow standard network connection practices to get up and running. Communications on Ethernet is then as simple as programming a communications rung into a ladder program in the Programmable Controller. Default values are set up for the Ethernet parameters, but can be altered by a user to help optimize Ethernet communications. Section 15.6 explains how to set these parameters and interpret performance data to further fine tune the Ethernet network.

The topics discussed here may be different for communications on ports 1 and 2 of the Model 650. Consult the appropriate SY/NET instruction bulletin for information pertaining to SY/NET and/or point-point SY/MAX communications.

The SY/MAX Model 650 processor has the ability to directly connect to a ThinWire Ethernet Local Area Network (LAN) and communicate with up to 29 other SY/MAX Ethernet compatible devices, with unique drop numbers. Through the use of Ethernet repeaters, this number can be expanded to up to 99 other devices, for a total of 100 SY/MAX devices.

The connection is accomplished with standard ThinWire cables, terminators and connectors. A unique SY/MAX address (drop #) on Ethernet is assigned to each Model 650 using a small rotary switch and 4 DIP switches located at the rear of its cabinet.

15.2 Hardware

All processors sharing the same Ethernet segment must be located within 607 feet (185 meters) of each other, as illustrated in Figure 15.1, unless repeaters are used.

The following hardware is required to connect each Model 650 processor in any size Ethernet configuration (also refer to Table 15.1):

- CABLE - Standard RG58 A/U or RG58 C/U coaxial cable, 0.2" diameter, 50-ohm, single shield. Belden® Type 8219, 8259, or equivalent.

- TEE CONNECTOR - Square D Class 8030 Type CCK-612. One of these is required for each 650 on the network. BNC Tee connector with jack-plug-jack (FMF) arrangement.

- TERMINATOR (50 ohm) - Square D Class 8030 Type CCK-613 (contains qty. 2). Two terminators are required, one for each end of the network.

- PRE-ASSEMBLED CABLE - Square D Class 8030 Type CC-601, 20 inches long (1/2 meter). This is the only pre-assembled cable available from Square D.

NOTE: *The 20 inch (1/2 meter) cable is the mandatory MINIMUM cable length between processors. Each Ethernet ThinWire segment may not exceed 607 ft.(185m); the total end-to-end length of the link (with repeaters) may not exceed 3,034 feet (925 meters). Refer to Appendix D for information on cabling other than Thinwire.*

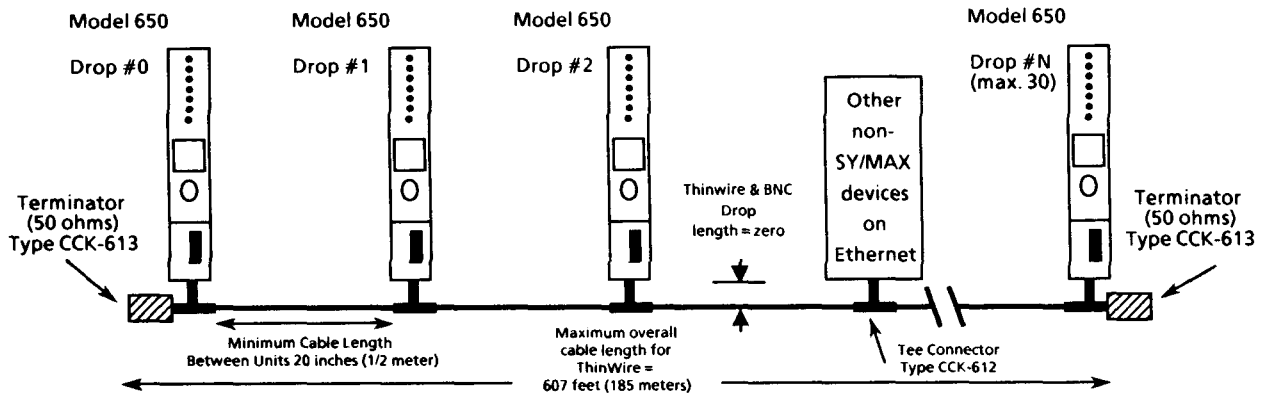


Figure 15.1 Possible Ethernet Configuration

ITEM	DESCRIPTION
Cable	RG58 A/U or RG58 C/U single shielded, 50Ω cable with BNC plug connectors (Belden 8219, 8259, or equivalent).
Tee Connector	*Class 8030 Type CCK-612* (AMP P/N 221543-2)
Terminators (2)	Class 8030 Type CCK-613 (50Ω, 1 watt BNC terminators - AMP P/N 221629-4, *contains qty 2*)
Cable End Connector	Class 8030 Type CCK-611 contains two male crimp-style BNC plug connectors (AMP P/N 227079-5)
Pre-assembled Cable	1/2 meter length, Class 8030 Type CC-601. Other cable lengths must be assembled by the user.

Table 15.1 Ethernet Hardware

- CABLE ENDS - Square D Class 8030 Type CCK-611 contains two male crimp-style BNC plugs. Cable and crimp tool (AMP CERTI-CRIMP Hand Tool P/N 220187-1) must be purchased separately.

NOTE: *It is extremely important that cable connections within the Ethernet be solidly made. Cold solder joints, poor connections, and other assembly flaws can cause intermittent communication problems. For highest reliability, all connectors should have gold-plated conductors. Also, avoid crushing the cable, or placing sharp bends in it.*

CONSIDERATIONS

- For proper operation, the LAN to which the Model 650 is connected must conform to all IEEE 802.3 specifications.
- See Appendix D for more Ethernet considerations.
- Model 650s may be installed in any register slot in any SY/MAX rack. While multiple Model 650s may be installed in the same rack, only the processor in slot 1 can directly control I/O.
- Each Model 650 draws 5,500 mA. While a single rack can hold multiple Model 650s, the power supply must be properly sized to handle the large load imposed by multiple processors. Loading restrictions preclude powering more than 3 Model 650s from the same power supply.
- For future expansion, empty tee or barrel connectors can be inserted at minimum intervals of 1/2 meter in a cable.

- The tee connector must be directly attached to the BNC connector on the Model 650, i.e., no "dropline" length is allowed between the tee connector and the processor.

- When adding processors to a functioning Ethernet, the cable connection should be made while the new member is powered down. In this way, network members can be added or removed from the Ethernet without powering down the link or interrupting network operation.

- Ensure the rubber insulating "boots" provided with the cable ends and terminators are installed. These are to prevent the link from being unintentionally grounded by accidentally coming into contact with the rack chassis or other means of current conduction.

- The Ethernet circuitry and cable is isolated from processor digital circuitry (300VDC, transformer-coupled). While bonding the cable conductors, tee connectors, or terminators to chassis/earth ground is not required for proper operation of the link, local electrical code requirements must be observed when laying out the installation. **At no time, however, should the link be grounded in more than one location.**

- To ensure reliable communications, it is recommended that all connectors have gold-plated conductors.

- The Ethernet cable should not be included in the same tray or conduit with power wiring, and should be routed so as to avoid passing in close proximity to electrically noisy devices such as power transformers, arc welders, contactors, etc.

15.3 Switch Settings

Refer to Figure 15.3. The Model 650 has a sixteen-position rotary switch and a four DIP switch bank for configuring the Ethernet. Accessible from the rear of the processor, both switches are used to encode a unique drop number from 0-99. Switch functions are described in the following sections.

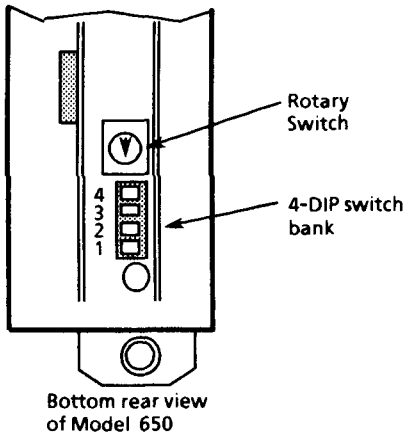


Figure 15.3 Rotary and DIP Switch Locations

CAUTION

Each device attached to Ethernet MUST be selected for a *unique* DROP NUMBER or the network will not operate properly.

15.3.1 LEAST SIGNIFICANT DIGIT

The decimal rotary switch (see Figure 15.3) establishes the least significant digit (LSD) of the SY/MAX drop number on Ethernet. Each rotary switch setting (0 to 9) corresponds to a unique SY/MAX drop number on Ethernet ranging from 0 to 9. Other switch settings are invalid. Refer to Table 15.2.

NOTE: *Use care when adjusting the rotary switch, since the separation between positions is very small. The data field of register 8093 reports the selected rotary switch position (in addition to the DIP switch settings). If the contents of register 8093 do not reflect the apparent switch position, rotate the switch one full turn to "wipe" the contacts.*

SWITCH SETTING	LSD
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9

Table 15.2 Rotary Switch LSD Setting

15.3.2 MOST SIGNIFICANT DIGIT

Immediately below the rotary switch is a DIP switch bank containing four switches. The four switches encode the high or most significant digit (MSD) digit of the drop number. The 4-bit binary code must be between 0 and 9 for network operation; other settings are invalid. Refer to Figure 15.4 and Table 15.3 for MSD switch settings.

MOST SIGNIFICANT DIGIT SWITCH SETTING*

1	2	3	4	MSD
Open	Open	Open	Open	0
Open	Open	Open	Closed	1
Open	Open	Closed	Open	2
Open	Open	Closed	Closed	3
Open	Closed	Open	Open	4
Open	Closed	Open	Closed	5
Open	Closed	Closed	Open	6
Open	Closed	Closed	Closed	7
Closed	Open	Open	Open	8
Closed	Open	Open	Closed	9
Closed	Closed	Closed	Closed	Disabled

Table 15.3

* Switch settings other than those in Table 15.3 are invalid.

After the drop number is set on the rotary and DIP switches, record the number on the front panel label for visual SY/MAX drop number on Ethernet identification, as shown in Figure 15.5.

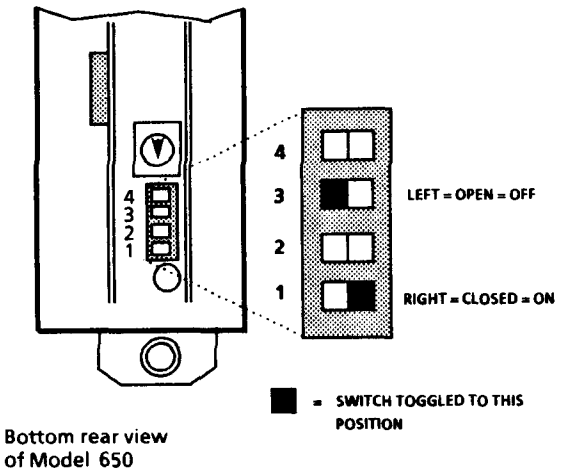


Figure 15.4 DIP Switch Identification

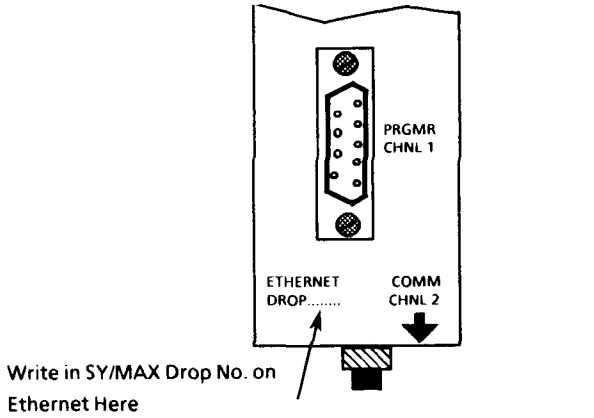


Figure 15.5 Location of SY/MAX Drop Number on Ethernet

NOTE: *The DIP switches have been set at the factory to all 1's, indicating non-Ethernet mode. The rotary switch factory setting will vary, but its setting will not affect the drop number, in this mode. Refer to Section 15.7 for further error message information.*

15.4 Defining an Ethernet Network with SY/MAX Devices

15.4.1 SY/MAX DROP NUMBERS

Each SY/MAX device on Ethernet must contain a valid drop number, in order to communicate with other SY/MAX devices on Ethernet. The rotary switch and 4 dip switches select the SY/MAX drop number for each device on Ethernet. SY/MAX devices must be numbered 0-99. See Section 15.3 for a complete explanation on valid and invalid switch settings.

NOTE: *All SY/MAX drop numbers on Ethernet MUST BE UNIQUE.*

15.4.2 ROUTING FOR PORT 3 (Ethernet port)

In order to communicate with other SY/MAX devices on the network, the drop number of the originating (source) device and the drop number of the destination (target) device must be known. In some situations, additional drop numbers must also be known. Refer to Appendix E for additional Ethernet routing information.

How to Use Routing

This section consists of three sample access routes from the Control Processor and Ethernet NIM portions of a Model 650. Refer to Figure 15.9 for a map of the access routes. Also, knowledge of variable route statements from Section 5.5 is assumed. Routing information is programmed into the Communication rung (READ, WRITE or ALARM) of the device initiating communications.

- **Example 1--Control Processor to Control Processor**

Figure 15.6 shows drop #36 reading control processor (image table) register 1001 of drop #84 into its own control processor register 2001.

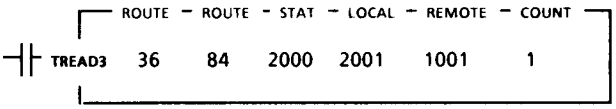


Figure 15.6 Rung for reading one image table register into another on Ethernet Port 3.

- **Example 2--Control Processor to another Processor's ENIM**

Figure 15.7 consists of drop #36 reading drop #84's Ethernet NIM storage (mailbox) register 3001 into drop #36's control processor (image table) register 2101.

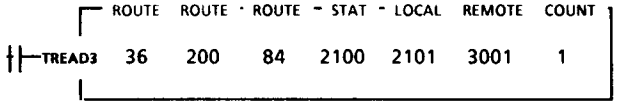


Figure 15.7 Rung for reading a mailbox register into an image table register on Ethernet Port 3.

- **Example 3--Control Processor to its own ENIM**

Figure 15.8 writes drop #36's control processor (image table) registers 4001 to 4128 into its own Ethernet NIM storage (mailbox) registers 5001 to 5128.

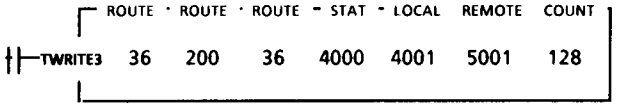


Figure 15.8 Rung for writing an image table register into its own mailbox register.

NOTE: *Routes 202, 203 and 233-254 apply only to ports 1 & 2, not to Ethernet port 3. Further information is available in Instruction Bulletin 30598-365-01A1, Multi-media Network Interface Module or 30598-257-02, Network Interface Module Instruction Bulletins.*

Route	Type	Port		
		1	2	3
200	Access NIM/ENIM registers	X	X	X
201	"whoever I am" (first route)	X	X	X
202	Port-to-Port (in-out same module)	X	X	N/A
203	Module Pair (last route)	X	X	N/A
204	Don't care - no action taken (ignored)	X	X	X
205	Variable Route	X	X	X
206-232	Not Used	N/A	N/A	N/A
233-254	Broadcast Capability	X	X	N/A

Table 15.4 2XX Routing Types by Port

When your Model 650 is received from the factory, *the switch settings will be configured for non-*Ethernet mode. This mode allows a user to power up the Model 650 and use it as a Programmable Controller control processor, but will NOT allow Ethernet communications. Non-Ethernet mode is indicated by all four dip switches set down (all ones), regardless of the rotary switch position. Any switch settings other than valid drop numbers (0-99) or Non-Ethernet mode will result in an error 918, illegal drop number.

NOTE: *If the Model 650 is being used to communicate on Ethernet via port 3 and on SY/NET via connection to a NIM on ports 1 or 2, the Model 650 may represent a different SY/MAX device drop number on each network. The Ethernet drop number is determined by the switch settings on the Model 650 while the SY/NET drop number is determined by the switch settings on the NIM. It may therefore be convenient to coordinate the two networks when possible to use the same drop number for both Ethernet and SY/NET networks.*

15.5 *Registers*

This section lists the Model 650 registers related to Ethernet. Figure 15.9 shows a spatial representation of the Control Processor, Ethernet NIM and Channels 1, 2 and 3 in the context of the Model 650.

15.5.1 MODEL 650 REGISTER OVERVIEW

Tables 15.5 and 15.6 outline the function of all Model 650 Control Processor and Ethernet NIM registers. Specific information about Control Processor registers can be found in Section 13.

15.5.2 CONTROL PROCESSOR REGISTERS

The Control Processor portion of the Model 650 consists of two parts. Registers 1-8000 are standard SY/MAX read/write registers, and registers 8001-8192 are SY/MAX control registers with several registers added for Ethernet considerations. Refer to Section 13 for specific control register information.

REGISTER RANGE	CONTROL PROCESSOR FUNCTION	TYPE R/W: Read/Write R: Read *Only*
1-8000	Processor registers	R/W
8001-8178	Model 650 control registers (see Section 13).	R/W
8179-8192	Model 650 control registers (see Section 13).	R

Table 15.5 Model 650 Control Processor Register Use

15.5.3 ETHERNET NIM REGISTERS

The Ethernet NIM (ENIM) portion of the Model 650 contains another 8192 registers. See Section 15.4.2 for examples of how to access the ENIM registers.

Registers 1-10 are "read-only" ENIM status registers:

- **Register 1** - has the device type (i.e. Model 650) stored in its data field.
- **Register 2** - has the Ethernet NIM processor revision number within it. This register contains the ENIM processor revision number in decimal. For example, a Revision 3.30 ENIM processor will have 0330D in this field.
- **Register 3** - contains the SY/MAX drop number on Ethernet, indicated by the DIP and rotary switches.
- **Registers 4-10** - (currently not used) have zeros in them and are read only.

Registers 11-2999 are not accessible. Reading or writing to these registers will yield a communications status register error 03, illegal address.

Registers 3000-6999 are user-defined standard read/write SY/MAX storage registers.

Registers 7000-7999 are user-selectable read/write registers for storing Ethernet performance parameters and data. The following are the Ethernet network parameters that can be accessed from read/write registers 7000-7999 (note that XX is the address or drop number of the receiving SY/MAX device on Ethernet):

- **7XX0** - Timeout time = the amount of time, in 10msec units, that each SY/MAX device will wait before assuming that no ACK has been received.
- **7XX1** - Total number of retries = number of communication retries (in addition to the original communication attempt) attempted with each SY/MAX device before declaring an error.
- **7XX2** - Average time to complete a SY/MAX transaction on Ethernet with each indicated drop number.
- **7XX3** - Total number of retries for each 1024 attempted transactions (milli-retries). This number divided by 1024 yields the average number of retries per SY/MAX transaction attempted.
- **7XX4** - Maximum time to complete a transaction (maximum time it takes to send a COMMAND or REPLY and receive an ACK).
- **7XX5** - Maximum number of retries to complete a transaction.
- **7XX6** - **7XX9** - Reserved for future use.

Registers 8000-8192 are not accessible. Reading or writing to these registers will yield a communications status register error 03, illegal address.

NOTE: *The Rack Address CLEAR ALL operation clears or resets all processor registers. However, the Ethernet NIM registers are not cleared. Those registers not cleared or reset include the ENIM storage (mailbox) registers (regs. 3000-6999) and the Ethernet communication performance data registers (regs. 7000-7999). Two alternatives for clearing or resetting the ENIM registers exist (Refer to Section 6.8 for more information on CLR ALL):*

1. *Remove the Model 650 from the SY/MAX rack, remove the battery form the Model 650 and wait for approximately 2 minutes for the memory to clear. Then replace the battery, place the Model 650 in the rack and provide power.*

2. *The second method is to program a series of WRITE rungs into the Programmable Controller to clear the ENIM storage registers (3000-6999). Cycling power then clears the performance calculations (7XX2-7XX5), but not the performance parameters (7XX0-7XX1) or the storage registers. Developing a small utility ladder program to clear the storage registers and set the default Ethernet parameters may be useful. This program could then be run anytime a Rack Address CLEAR ALL is issued to clear or reset all the desired registers.*

REGISTER RANGE	ETHERNET NIM FUNCTION	TYPE R/W: Read/Write R: Read Only
1-10	NIM status registers	R
11-2999	Reserved for future use.	N/A
3000-6999	*NIM Storage registers*	*R/W*
7000-7999	Contains Ethernet network parameters.	R/W
8000-8192	Reserved for future use.	N/A

Table 15.6 Ethernet NIM Register Use

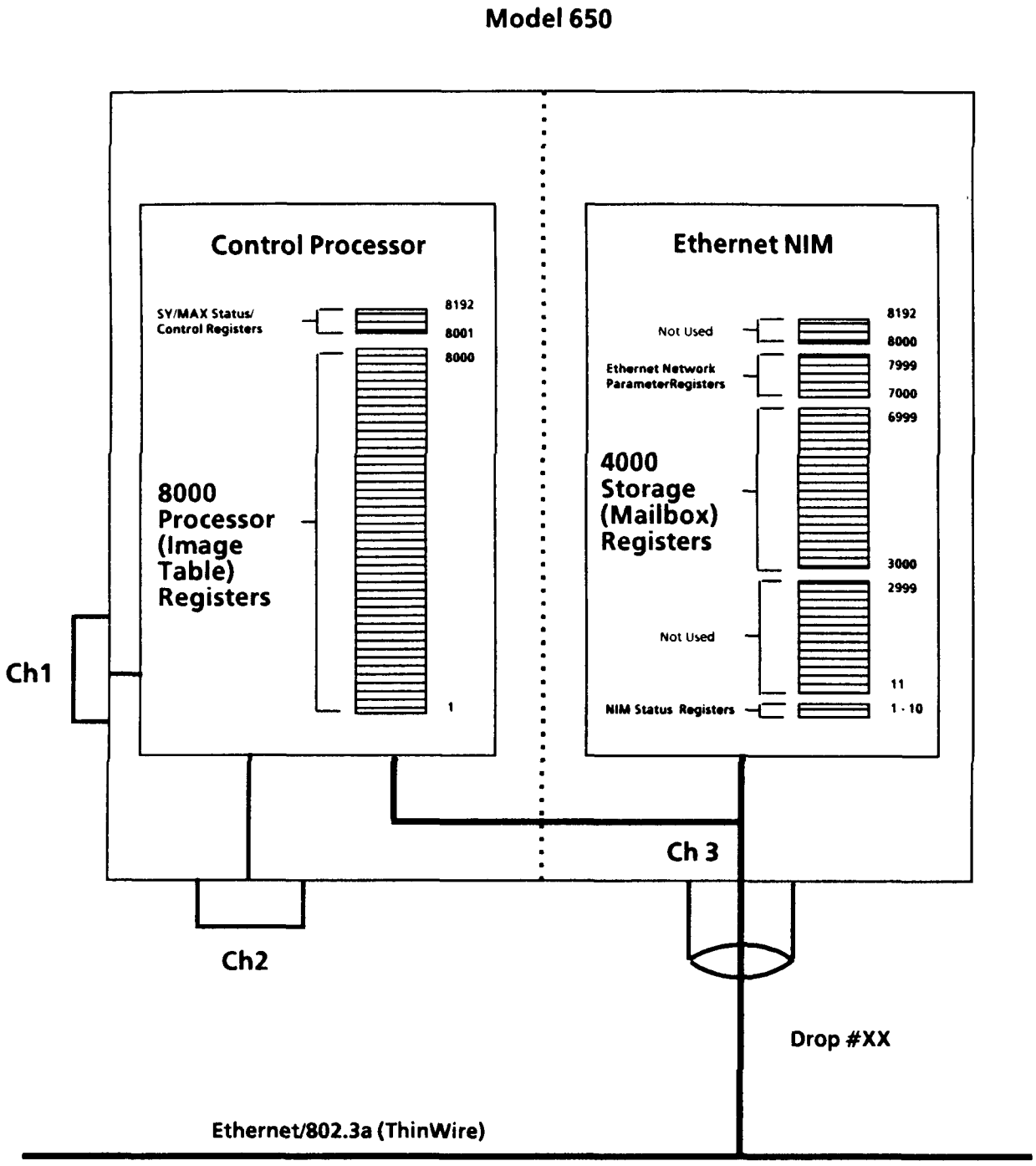


Figure 15.9 Model 650 Registers

15.6 Ethernet Communication Parameters

15.6.1 OVERVIEW

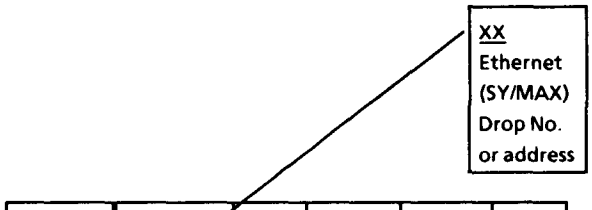
NOTE: *Again ensure that all SY/MAX on Ethernet devices have a unique SY/MAX address ranging from 0 to 99.*

1000 registers have been set aside in the ENIM of the Model 650 Programmable Controller to store Ethernet communication parameters for communicating with up to 99 other SY/MAX devices (100 total SY/MAX devices) on Ethernet. The distinction "SY/MAX" devices is made because other non-SY/MAX devices may be on Ethernet, but have their own unique Ethernet address and will not affect SY/MAX or be affected by non-SY/MAX products on the network. Thus, the total number of devices on Ethernet is only limited by Ethernet physical and electrical limitations, while the total SY/MAX limit is 100 devices, providing other requirements are met also. These 1000 ENIM parameter registers are numbered 7000-7999 and are Read/Write registers. They are organized in groups of 10 registers for each of the 100 SY/MAX drops on Ethernet, although only 6 of these 10 registers (7XX0-7XX5) are currently being utilized, with the 4 remaining registers (7XX6-7XX9) reserved for future product enhancements to the Model 650. Avoid using these 4 reserved registers for storage, so as to maintain compatibility with future releases of this product. This helps preserve ladder program and operational upward compatibility.

The registers are organized by the SY/MAX Ethernet drop number. For SY/MAX address 00 (drop 00), the registers start at 7000 and end at 7009. For SY/MAX address 01 (drop 01), the registers start at 7010 and end at 7019 (and so on up to 7999). Thus the 1000 registers divided by 10 registers per drop number = 100 drops (the maximum number of devices of any combination of Model 650's, VAX computers and other SY/MAX devices). The middle two numbers of the register indicate which drop that register is associated with. For example, register 7010 is associated with SY/MAX drop 01. The shorthand notation for this encoding method is 7XX0, where XX indicates the SY/MAX drop number.

15.6.2 SETTING ETHERNET COMMUNICATION PARAMETERS (7XX0 and 7XX1)

Since the Ethernet Local Area Network (LAN) is a non-deterministic network, there are no guarantees as to when or if you can begin communication on the LAN. The first two registers for each drop, 7XX0 and 7XX1, allow the user/programmer to establish the length of the Ethernet timeout and the number of retries, respectively, when the Programmable Controller is communicating (e.g. via READ or WRITE rung for port 3) with a device on Ethernet with SY/MAX drop number XX. Refer to Table 15.7. The timeout register, 7XX0, indicates how many 10 msec units to wait before it is assumed that no ACK has been received from a remote device and attempt another retry. The total number of retries (in addition to the original communication attempt) is indicated in the retry register, 7XX1. Upon power up of the Model 650, the 7XX0 registers default to a value of 100. This means 100 10msec units (which is equivalent to 1 second) is the time the Ethernet processor in the Model 650 will wait before attempting to retransmit the COMMAND. Register 7XX1 defaults to 2 retries, before posting an error 17, indicating a remote device is inactive. However, a valid ACK and retry can still occur in spite of the fact that an error message has been posted.



Register Address Range	Parameter	Units	Default	Max.	Min.
7XX0	Timeout Time	10ms	100	65536	1
7XX1	No. of Retries	1 retry	2	255	0

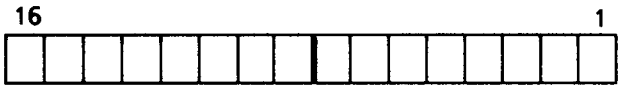
Table 15.7 Timeout Time and Retries

7XX0 Timeout time

is the amount of time in 10 msec units, each SY/MAX device waits before assuming a message was lost (no acknowledgment returned on Ethernet). The default is 100 units or 1 second.

# of timeout units X 10 msec = total time

e.g., 100 timeout units X 10 msec=1 sec. Timeout time may be set for a maximum of 65535 units X 10 msec or ≈ 11 minutes. The minimum is 1 unit or 10 msec. If the timeout is set to 0, it defaults to 1. The timeout time for that address or drop number being communicated to is set up in the initiating device's 7XX0 register (not the receiver's 7XX0 register).

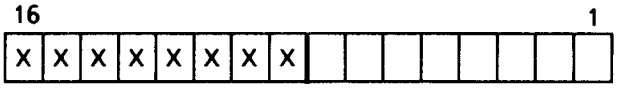


7XX0--Full 16 Bits used as unsigned integer

Timeout - 65535 is the maximum value (full 16 bits used) a user can place in this register. This is treated as an unsigned integer. Any attempts to use a negative number (signed integer) will result in the corresponding unsigned integer being used. If a zero (0) is placed in this register, the system will use a count of 1 for actual communications. The zero (0) will still remain in the register and will appear in any attempt to READ this register, but the value of 1 will actually be used for communication. Thus, with the above exceptions, there is no explicit range checking on the data put in by the user.

7XX1 Number of retries

is the number of attempts by a Model 650 to communicate with other SY/MAX devices before indicating an error. Retries keep occurring until a positive Acknowledgment is returned over Ethernet. The number of retries defaults to 2, has a maximum of 255 and a minimum of 0. The number of retries for the address or drop number being communicated to are set up in the initiating device's 7XX1 register.



7XX1--Lower 8 Bits used as unsigned integer

X = Don't care bits

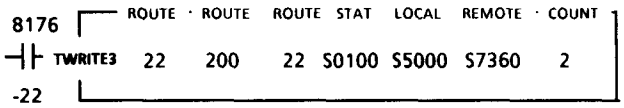
255 retries is the maximum value (lower 8 bits of register used only) used by the Programmable Controller for Ethernet communications. Anything greater than 255 put in the register will result in the high byte of 7XX1 being truncated and the lower byte used for actual communications, although the full 16-bit value will remain in the register. Similiarly, if a negative number is used, the number of retries corresponding to the lowest byte (high byte truncated) will result. A value of zero (0) can be legitimately used, resulting in no retries (only the original communication attempted). However, this is usually not a practical parameter, since the Ethernet network may be busy. Thus, there is no explicit range checking on the data put in by the user.

Timeout and Retry Example

Both the 7XX0 and the 7XX1 registers for SY/MAX drop #22 can be changed by using the following rungs in a ladder program:

NOTE: *Bit 8176-22 may be used to set up new Ethernet parameters prior to program execution. This is made possible by using the second dummy scan of the control processor to set up the parameters. Adding the ladder rung below enables you to set up the parameters prior to program execution. This particular example sets up timeouts and retries in drop #36. Refer to Register 8176 (Primary Control Processor) in Section 13 for more information.*

Ladder Rung to Establish ENIM Preset Value



Ladder Rung to Transfer Preset Values to ENIM Register

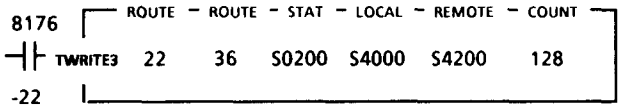


Figure 15.10 User-Modified Timeout and Retry Example Rungs

In this example, rungs from a Model 650 with a SY/MAX drop number of 22 on Ethernet will set up Ethernet parameters for communication with SY/MAX drop 36 on Ethernet. The first rung presets register S5000 = 55 (representing a timeout of 550 msec) and S5001 = 5 (for 5 retries). The second rung then writes from drop 22 to its own ENIM parameter registers 7360 and 7361. This sets up the Ethernet communication parameters for communicating with SY/MAX drop 36. Finally, at some later point in the program, the last rung shown writes from drop 22, registers 4000 through 4127, to drop 36, registers 4200 through 4327, using the Ethernet communication parameters established in registers 7360 (550 msec timeouts) and 7361 (5 retries).

Figure 15.11 represents 3 SY/MAX drops with 1000 read/write registers each (7000-7999) available for network parameters.

For instance, at the top center of Figure 15.11, the timeout time was left at the default of 100 (register 7220) for communications between drop #36 to drop #22. And the number of retries (2) is also left at the default set in register 7221. So the timeout time is 100 units (also the default) or 100 X 10 ms or 1 second. Register 7221 also specifies that the message will be transmitted once and then retried twice, if necessary, until it is acknowledged by drop # 22.

Registers 7230 to 7239 are available for when drop #36 communicates with drop #23; registers 7240 to 7249 are available for when drop #36 communicates with drop #24, and so on.

The box on the bottom left of Figure 15.11 represents drop #22's Ethernet network parameters. The timeout time of 55 is set in register 7360 for when drop #22 communicates with drop #36. The number of retries is set in register 7361 for drop #22 to retry another 5 times, if necessary, until it receives an acknowledgment. The remaining parameters in registers 7362 through 7369 contain transmission performance data. So all drops may have their 1000 registers (10 per drop) set up for each of the 100 drops. However, if values are not placed in the registers, they will default to the values shown in Table 15.7.

15.6.3 EVALUATING ETHERNET COMMUNICATION PERFORMANCE (7XX2 - 7XX5)

Registers 7XX2-7XX5 represent performance information for communicating with various SY/MAX devices on Ethernet. They provide a means of monitoring Ethernet communications and changing parameters in 7XX0 and 7XX1 based on these performance values. These registers are read/write registers, but are intended to be examined only, as data is calculated and updated by the Programmable Controller periodically and placed in these registers. Writing to these registers may only have a temporary effect, as new data is calculated dynamically and replaces previous data in these registers. These registers are only updated when communications (COMMANDs and REPLYs) are active with drop XX. The specific times at which they are updated are explained below.

NOTE: *Writing zeros to 7XX2-7XX3 does NOT have the effect of zeroing out calculations performed previously. However, writing zeros to 7XX4 and 7XX5 does zero out previously performed calculations.*

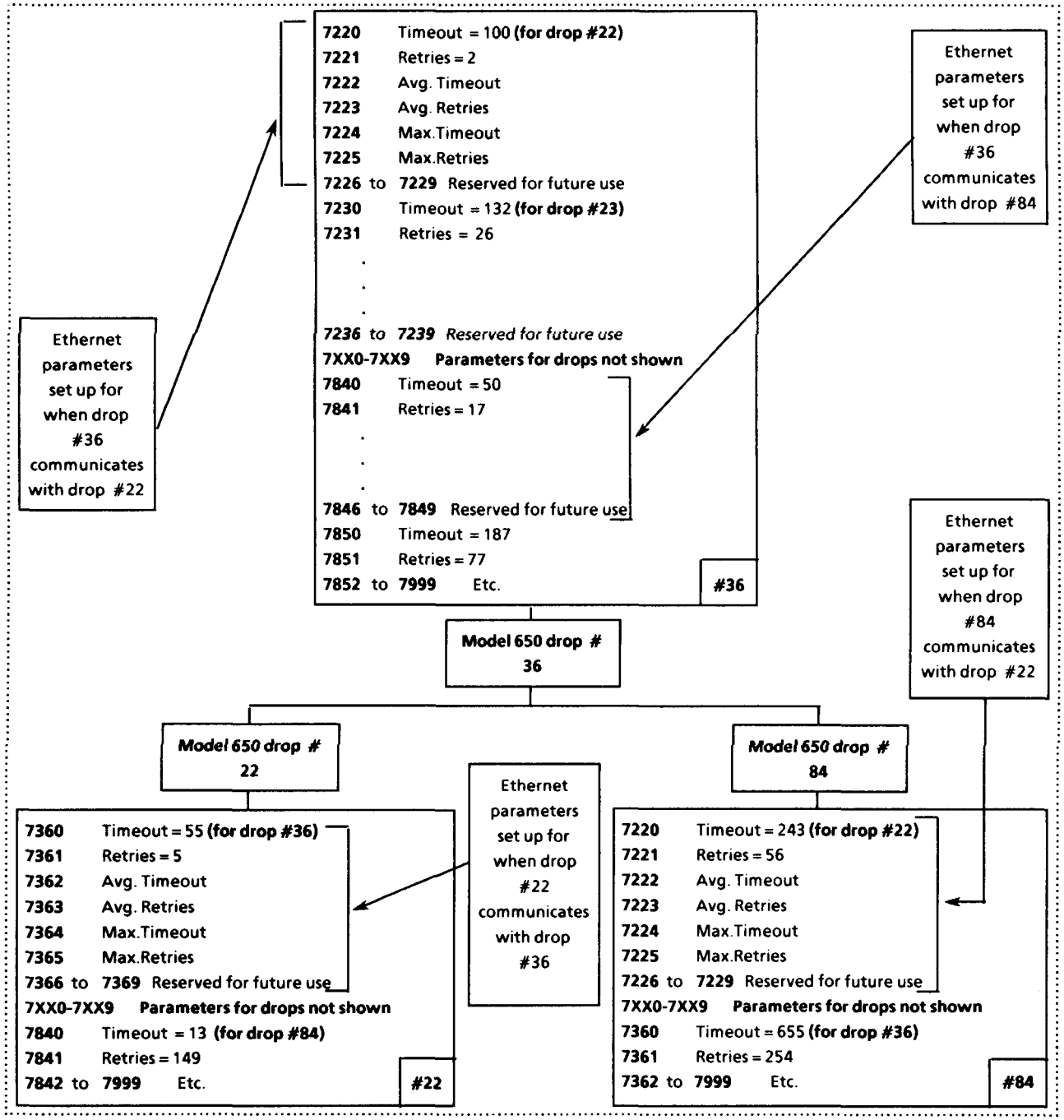


Figure 15.11 Examples of Ethernet parameters set up on Registers 7XX0-7XX9

As mentioned above, Registers 7XX2-7XX5 are ordinarily computed by the Ethernet NIM. Table 15.8 summarizes 7XX2-7XX5's communication performance parameters.

Register Address Range	Parameter	Units
7XX2	Avg. Time to complete a transaction	10 msec
7XX3	Avg. number of retries for each attempted transaction	Retries per 1024 attempted
7XX4	Maximum time to complete a transaction	10 msec
7XX5	Maximum number of retries for each attempted transaction	Retries per transaction

Table 15.8 Ethernet Communication Performance Parameters (7XX2-7XX5)

7XX2 Average time to complete a SY/MAX transaction on Ethernet

This is the average total time it takes to send a SY/MAX COMMAND or REPLY to drop XX and receive an ACK. All transmissions (the original plus the retries) are included in this calculation. Like 7XX0 and 7XX4, this is expressed in 10msec units.

7XX3 Total number of retries for each 1024 attempted transactions (milli-retries).

This number divided by 1024 yields the *average number of retries per* SY/MAX transaction attempted. This was done to preserve the accuracy of the average in an integer format.

7XX4 Maximum time to complete a transaction

This is the maximum time it takes to send a COMMAND or REPLY and receive an ACK. Assuming 7XX0 has not been decreased, 7XX4's value will be equal or less than 7XX0's value. 7XX4's value can be cleared to generate a new maximum time.

7XX5 Maximum number of retries for each attempted transaction.

This is the maximum number of retries that occurred during a SY/MAX COMMAND or REPLY transaction. Assuming that 7XX1's value was not decreased, 7XX5's value would be equal to or less than 7XX1's. Like 7XX4, 7XX5's value can be cleared to generate a new maximum.

The previous example (Figure 15.10, User-Modified Timeout and Retry Example Rungs) illustrates the process needed to set Ethernet communication parameters (7XX0 and 7XX1). This is a result of ladder programming packages having no access directly to monitor and edit data in the ENIM registers and must therefore do so through ladder programming. It may be helpful, if the required registers are not needed for other purposes, to map the active ENIM parameter registers into registers 7000-7999 of the processor registers. In reference to the example configuration in Figure 15.12, the following rungs in drop 36 could be used to map performance data for drops 22 and 84 into the processor image table registers.

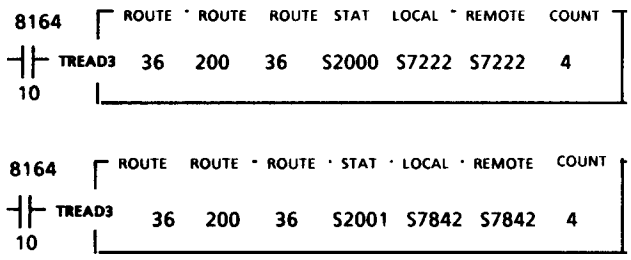


Figure 15.12 Example Ethernet Communication Performance Rungs

In this example 8164-10 is used to trigger the read rungs. Register 8164 is the Programmable Controller timer register and bit 10 is set approximately once every 2 seconds. Thus the performance data for drops #22 and #84 is updated about once every 2 seconds. Of course other methods could be used to trigger when the update actually takes place.

Now since the data is updated in the processor registers, it can be viewed via a ladder programming device on a data screen. It should be noted that to change the parameters (7XX0 or 7XX1), the appropriate WRITE rungs still need to be added to the ladder program, as the ENIM registers cannot be changed via a ladder programming device.

The previous two examples allow the user to set the communication parameters for drops 22 (7220 and 7221) and 84 (7840 and 7841) and then examine the performance (7222-7225 and 7842-7845) and make adjustments accordingly. For example, if the performance data (averages and maximums) are near the maximum value set by the user (as indicated in 7220-7221 and 7840-7841), perhaps the timeout should be extended and/or the replies increased.

15.6.4 MISCELLANEOUS CONSIDERATIONS (7XX2-7XX5)

Upon power up, registers are set to zero (0) and updated under the following circumstances. The retry variables (7XX3 and 7XX5) are updated when:

1) an ACK is received in response to a command that is sent out
2) the maximum number of retries is met (timed out).

The time variables (7XX2 and 7XX4) are updated when:

1) an ACK is received,
2) a NAK is received
3) a timeout occurs
4) a transmission error is received.

All the above registers are cleared when the Ethernet communication port fails or the Programmable Controller is powered up.

If registers 7XX0 and/or 7XX1 were changed prior to any communications with drop XX or if the default values were used for communcications, then 7XX3 and 7XX5 should always be less than the value specified in 7XX1. It is possible however, that if 7XX1 was subsequently changed to a lower value after communications had been activated, that the previous running average (7XX3) and peak retries (7XX5) could be higher than the newly updated maximum user specified in 7XX1 (which is now set lower than before). Keep in mind that 7XX3 is in milli-retries, that is the number of retries for 1024 attempted transactions. Writing zeros to these registers for a specific drop after communications have already been active will NOT clear these calculations in 7XX2 and 7XX3. Thus, care must be exercised when interpreting these values if timeouts (7XX0) or retries (7XX1) have been changed after communications have been active for drop XX.

If ENIM registers 7XX0 and 7XX1 are changed while comms rungs are still waiting for ACKs to outstanding COMMANDS, the new values entered will be used in determining timeouts and retries. For example, assume initially 7XX0 is set to 200 (2 seconds) and 7XX1 is set to 10 retries, and 5 READ/WRITE rungs are solved true and sent to the communication buffers. If 7XX0 and 7XX1 are changed back to 100 and 2 respectively (the default values), the new values (1 second and 2 retries) will govern communications with drop XX.

Reserved Registers (Registers 7XX6-7XX9)

Registers 7XX6-7XX9 are not available at this time. These registers should not be used for storage purposes, so that upward compatibility with future SY/MAX devices may be maintained.

15.7 Ethernet Errors and Diagnostics

This section describes errors related to Ethernet only. For other Model 650 operational errors, refer to Appendix A. Any errors and/or diagnostic conditions which are related to Ethernet communications are posted in processor control register 8094. Ethernet Errors also post a value of 929 in processor control registers 8175 to indicate that an Ethernet error has occurred and informs the user to examine register 8094 for more specific information. Ethernet Diagnostic codes are posted to 8094 under various conditions, but do NOT post an error indicator in 8175 and usually do not require any action by the user to fix them. Refer to Section 15.7.2 for more explanation. They are primarily intended for informational purposes. Each group of codes has various effects on processor execution, Ethernet communications, and LED activity. Refer to Table 15.9 for various Ethernet LED and processor states by error code.

Error Codes	Ethernet LED	Port 3 Activity	Ladder Scan
<900	flashing	continuous	continuous
900-919	solid	halt	continuous
≥920	solid	halt	halt

Table 15.9 Register 8094 Ethernet Error Codes by LED and Processor

Processor LED states are described in Section 4.2. Depending on the nature of the Ethernet Comm Error, the status of the processor and/or the network is indicated.

15.7.1 ETHERNET ERROR CODES

Register 8175 represents the starting point for all SY/MAX error code investigations, including Ethernet errors. In those cases where register 8175 contains error code 929, register 8094 will contain additional information about the nature of the Ethernet failure. Table 15.10 lists Ethernet-related errors which may be found in register 8175.

ERROR CODE	DESCRIPTION
929	Ethernet communication error, refer to register 8094.
979	Processor installed in slot 1 with rack addressing missing
985	Processor is not installed in slot 1 though rack addressing is present.
30600	Handshake failure between control processor and communication processor.

Table 15.10 SY/MAX-Ethernet Errors in Register 8175

Error 985 or 979 typically occurs when using multiple Model 650s in the same rack.

Model 650 register 8094 is used to store communication error codes related to error 929. Table 15.11 shows error codes that may appear in register 8094, along with a brief description of each.

ERROR CODE	DESCRIPTION
916	Ethernet Connector Problem
917	Duplicate Drop Number*
918	Illegal Drop Number*
919	General Hardware Failure
920	Local RAM READ-after-WRITE test failure.
921	Local RAM destructive READ-after-WRITE test failure.
922	Shared RAM READ-after-WRITE test failure.
923	Shared RAM destructive READ-after-WRITE test failure.
924	Link controller initialization test failure.
927	Link controller run-time failure.
30500	Bus signal error due to time out.
30600	Handshake failure between control processor and communication processor.
30800	Software diagnostic error.
30900	Parity error.
31100	Ethernet NIM PROM Checksum error.
32700	Main processor failure.

Table 15.11 Ethernet Errors in Register 8094

* *Check DIP and/or rotary switch settings and examine register 8093.*

15.7.2 ETHERNET DIAGNOSTIC CODES

There are several conditions which may occur on Ethernet which post diagnostic codes in the processor control register 8094. Refer to Table 15.12. Unlike the error codes in section 15.7.1, there will be NO corresponding error code posted in 8175 to indicate a diagnostic code in 8094. The Ethernet network LED will blink when a diagnostic error code is occurring. These codes remain in 8094 until another diagnostic condition occurs and the proper code replaces the previous one. These codes result from transient errors and, unless they occur frequently, do not require any action by the user. Both the processor and Ethernet communications will continue as usual. These codes may prove useful in determining network problems if any of these codes occur repeatedly and/or often. The following diagnostic codes reference errors detected by the Local Area Network Contoller for Ethernet (LANCE) chip used in the Model 650.Additional troubleshooting insights can be found in Appendix A. If any of these error codes occurs repeatedly, an Ethernet 802.3 protocol analyzer should be used to troubleshoot and determine which device(s) on Ethernet may be sending and/or receiving improper Ethernet packets.

DIAGNOSTIC CODE	DESCRIPTION	SOLUTION
800	Unable to connect to Ethernet. This error occurs when the processor is set to a valid Ethernet SY/MAX drop number, but is unable to initially connect to the network. Probably due to an improper connection or termination of the network cable. This is similar to error 916, the difference being that error 916 occurs after Ethernet communications have already been active, whereas diagnostic 800 occurs after power-up but before any communications are initiated.	Connect and/or check cable and cable terminations.
801	LANCE-Collision. No heartbeat collision after packet sent by this device.	No user action required. Possible faulty device.
802	LANCE-Missed packet. This error indicates that an Ethernet packet intended for a device was missed by that device due to all buffers currently being used.	Transmitting device will probably retransmit packet. Analyze traffic patterns on network.
803	LANCE-Improper frame size. This error indicates that the incoming packet contains a non-integer multiple of 8 bits AND there was an incorrect CRC.	Ethernet packet is discarded. Analyze Ethernet packets.
804	LANCE-Buffer Overflow condition. This error indicates that the LANCE has overflowed internal LANCE buffers to service incoming messages.	No user action required. Ethernet packet is discarded. Jabber-related error.
805	LANCE-Incorrect CRC detected. This error indicates the Cyclic Redundancy Check (CRC) code in the Ethernet packet does not match the CRC calculated for the data contained in the packet.	No user action required. Ethernet packet is discarded. Due to noise, faulty or improper cable connection or incorrect packet transmission/reception.
806	LANCE-Doesn't own any more buffers. This error indicates that the LANCE cannot store any more of the packet due to no available buffers.	No user action required. Ethernet packet is discarded. Jabber-related error.
807	LANCE-End of Ethernet Packet not detected. An Ethernet packet larger than the maximum allowed was received. This type of packet is called a jabber.	Check cable and terminator on the network. Ethernet packet is discarded.
808	LANCE-Late collision on Ethernet. A collision has occurred on Ethernet after the time allowed for a 'normal' collision. This does not cause the LANCE to retry any messages.	Check cable and terminators on the network. Ensure that distance specifications have not been violated.
809	LANCE-Loss of carrier detected. The LANCE cannot detect the Ethernet carrier on the network.	Check cable and terminations on the network.
810	LANCE-Maximum number of collisions occurred. 16 collisions have occurred on the network while trying to transmit a message. This may cause a RETRY of the message.	Check cable and terminations on the network. Analyze traffic volume on the network.
811	Unable to communicate on Ethernet. This error occurs when a processor connected to the network and having a valid Ethernet SY/MAX drop number is no longer able to communicate on the network. Probable cause is a loose or disconnected "T" connector at the Model 650 Processor. In pre-Rev 1.21 processors this condition was indicated by error code 916 and required that the error condition be removed and power be cycled to the processor in order to resume Ethernet communications.	Connect and/or check Ethernet "T" connector, cable and cable terminations. Reconnecting the "T" connector should allow resumption of Ethernet communications without the need to cycle power to the processor.

Table 15.12 Ethernet Diagnostic Codes in Register 8094

Troubleshooting Notes:

1. *Packet errors can be analyzed with a LAN 802.3 protocol analyzer.*
2. *Cable, connector and terminator errors can be analyzed by physical inspection and with a device with Time Domain Reflectometer (TDR) capabilities.*
3. *Noisy networks can be analyzed with oscilloscopes or spectrum analyzers.*

APPENDIX A ERROR CODES AND TROUBLESHOOTING

A.1 Introduction and Description

Errors detected by the processor, and error codes displayed on a CRT programmer, provide information that is useful for isolating problems.

Some errors appear as self-explanatory messages on the programmer's screen. Other types of errors involve error *code numbers*. This appendix deals only with the errors that appear as codes.

Error code numbers can be separated into two general categories which appear in different locations on the programming equipment's screen. The two categories are **Peripheral to Programmable Controller System Interaction Errors** and **Programmable Controller System Operational Errors.**

Peripheral to Programmable Controller Interaction Errors are caused either by an attempted illegal operation with the Model 650 or by malfunctioning communication hardware. These errors comprise three types of error codes, and are displayed at the *bottom* of the programming equipment's screen. In some special cases, these codes appear in the status register of a communications rung. See Section A.2 for details.

Programmable Controller System operational errors are errors detected *within* the Model 650 or in any of the modules in the overall programmable controller system. These errors are displayed in the STATUS mode of the programming equipment next to a "ERROR NUMBER" message. The number displayed is also the number contained in the Model 650's Control Register 8175, or possibly in a Local Interface Module's error code control register (when using remote I/O).

Normally when correct programmable controller system operation is restored, the error code is cleared. The Model 650 contains a valuable troubleshooting feature that allows a user to store up to 16 of the most recent error codes in a user-defined register block. See Section 13.2, "Control Registers" (register 8107).

A.2 Peripheral-to-Programmable Controller System Interaction Errors

These types of errors are caused either by attempts to perform illegal operations with the Model 650 or by improperly functioning communication hardware. This type of error is generated only when using programming equipment to perform Model 650 operations, or when communication between the Model 650 and another device is interrupted.

The three types of peripheral-to-programmable controller interaction error codes are **Processor Errors, Transmission Errors** and **Tape Errors.** Each is displayed as a message combined with a code on the programmer's screen, and is described in Sections A.2.1 to A.2.3. If a peripheral-to-programmable controller error occurs while a programmer is connected to the Model 650, one of the three messages (*processor, transmission,* or *tape* error) is shown on the display along with a number, such as:

TRANSMISSION ERROR 74

If an error occurs while the Model 650 is executing a communication rung, a *processor error* is generated. The error code number also appears in the status register of the communication rung which failed to execute properly.

A.2.1 PROCESSOR ERRORS

These errors indicate that an attempted operation was not successfully completed. Error code numbers for processor errors range from 01 to 99. To isolate the error, find the programmer-displayed code number in the table of Figure A.1. Use the CLEAR/DEL key to clear the error indication, and then take appropriate corrective action.

NOTE: *Error codes indicated by an asterisk (*) in the following Error Code Tables are displayed in the status register of a communications rung.*

ERROR #	DESCRIPTION OF PROCESSOR ERROR
00	No Error
01*	Illegal protocol opcode - not a valid Model 650 instruction. Re-enter the instruction.
02	Illegal intermediate code format. Re-enter the desired operation.
03*	Illegal address attempted.
04	Illegal rack addressing
05*	Instruction not allowed in this type of processor.
06	Item searched for was not found
07*	An attempt to alter data in a protected register has been made. Check Control Register 8176 or 8178.
08	An attempt to access protected memory has been made. Check Control Register 8176 or 8178.
09*	An attempt to alter a read-only register, or a register containing external inputs, has been made.
10	An attempt to exceed memory limitations has been made.
11*	Communication error - receiver overflow. Re-enter the desired operation.
12	Illegal CPU rack addressing - register assignments must be divisible by four.
13*	Communication error - link error. Check Control Register 8175, the status register of the communications rung, or the cabling between the Model 650 and attached devices.
14	The operation entered is not allowed while the processor is running.

Figure A.1 Processor Error Codes

ERROR #	DESCRIPTION
15*	Communication overflow. Check the baud rate for compatibility between devices, and re-enter the operation.
16	The register count in a communication rung is too large for this type of processor. Reduce the number of registers.
17*	Remote device inactive (this error is generated by the Network Interface Module - check cabling between devices).
18	Illegal rung number. Re-enter.
19*	Illegal READ parameter assigned. Re-enter.
20	Illegal, or no, channel number assigned. Re-enter.
21*	An attempt to change a forced register or write to an external output while the bus master is halted was made.
22	Forcing is inhibited at this time. Check Control Register 8176 or 8178, and keyswitch position.
23*	An attempt to alter data in a fenced register has been made.
24	An attempt to force an unforceable register was made.
25*	CPU error, check Control Register 8175.
26	Rack addressing/user-memory overlap. Redefine one or both.
27*	Memory error - a CLEAR ALL operation is needed to reset all data to the default state.
28	Illegal baud rate selected. Re-enter.
29*	An attempt to send a message with an illegal route was made. Re-assign route.
30	An attempt to alter memory protected by the inhibit coil was made.
31	Tape error - see Section A.2.2
32	Rung editing operation not allowed in protected memory.
33*	Tape error - see Section A.2.2
34	Not used
35*	Tape error - see Section A.2.2
36	"Replace Rung" operation not allowed on this rung. Use DELETE or INSERT operations only.
37*	Tape error - see Section A.2.2
38	Program view inhibit is in effect
39*	Alarm is already set within the D-LOG module
40	Hardware memory security is active

Figure A.1 Processor Error Codes (continued)

ERROR #	DESCRIPTION
41*	Illegal register WRITE into the D-LOG module. Some of the D-LOG registers will not accept a WRITE operation.
42	Illegal clock data.
43*	Illegal operation attempted - D-LOG module is *protected*.
44	The indicated file is not resident in memory.
45*	Operation not allowed - D-LOG tape operation is in progress.
46	An attempt has been made to access a protected file..
47*	Operation not allowed in present keyswitch position.
48	Programming a MCR in the subroutine area is not allowed.
49	I/O register or channel is safeguarded.
50	Rack addressing is not alterable while forcing is in effect.
51*	Module addressing error - module is not seated properly or is missing, no rack addressing is assigned to the module, or rack address exceeds limitations of the Model 650.
52	Illegal file type.
53*	Tape error - see Section A.2.2
54	Illegal MARK number.
55	Tape error - see Section A.2.2
56	Not used.
57	COMMS queue buffer forced emptied.
58	Not used.
59	Battery low.
60	Illegal MARK number - GOTO or GOSUB with numbers above 8190 are illegal.
61	MARK ST. SUB rung cannot be inserted - only an APPEND operation is allowed.
62	RTN rungs are not allowed in the mainline ladder program area.
63	This MARK number was previously used in the program. Each MARK must be assigned a unique number.
64	Operation not allowed while the Model 650 is running - cannot delete a RTN rung if any GOSUB rung contains the same number.
65	Cannot delete MARK ST. SUB rung without first deleting all subroutines.
66	Cannot delete a MARK rung without first deleting the associated RTN rung.

Figure A.1 Processor Error Codes (continued)

ERROR #	DESCRIPTION OF PROCESSOR ERROR
67	RTN rungs must have an associated MARK rung.
68	Enter only one RTN rung for each MARK number.
69	GOSUB rung cannot call a subroutine that has no RTN rung
70	GOTO rung cannot jump to a MARK reserved for a subroutine
71	Cannot delete a MARK rung without first deleting all GOSUB and GOTO rungs associated with it.
72	*Rung cannot have unused MARK number /* GOTO or GOSUB with an undefined MARK is not allowed / MARK rung must be programmed before the GOTO or GOSUB is programmed.
73	Not used
74	Processor error generated by a programming device--not by a processor.
75-78	Not used
79	Operation not allowed while the Timed Interrupt is enabled.
80	Operation cannot be initiated due to insufficient data. Too few parameters are indicated after a command word.
81	The command is not recognized. Examine for *validity*.
82	Illegal parameter. Re-check command syntax for valid numeric values, register addresses, register values, route values and filespec.
83	Illegal source parameter. Out of range or not a numeric value.
84	Illegal destination parameter. Out of range or not a numeric value.
85	Invalid filespec. Improper use of BUS port in command device, ID doesn't match device in fdp, or channel number above 5.
86	Invalid input channel. Improper use of BUS port in command device, ID doesn't match device in fdp, or channel number above 5.
87	*Invalid output channel. Improper use of* BUS port in command device, ID doesn't match device in fdp, or channel number above 5.
88	Device not available. Bubble 'dead' (no 12VDC, etc.).
89	A serial link cannot be established. No acknowledgement received.
90	Hardware read fault. Checksum mismatch.
91	Hardware write fault.
92	An attempt to read data failed. Software checksums mismatch outside of bubble/RAM disk.

Figure A.1 Processor Error Codes (continued)

ERROR #	DESCRIPTION
93	An attempt to write data failed.
94	An attempt to read from the directory has failed.
95	An attempt to write to the directory has failed.
96	*Space reserved for directory data is full.*
97	Device space is full. Bubble or RAM full.
98	The indicated file is not found in MCM memory.
99	The file has been restricted and may not be accessed. Attribute is limiting access.
100	File already exists.
101	The indicated file is already opened.
102	An error exists in the format of the file. Format doesn't match filetype.
103	The wrong type of file was indicated.
104	Open attempt to open file.
105	An external I/O attempted to unopen a file.
106	An attempt was made to write to a read only file.
107	Record out of range. Offset is outside of program size.
108	Too many files are open. The attempt to open another file exceeds MCM limitations.
109	*Operation aborted by user.*
110	A source filespec is required before the intended operation will initiate.
111	A destination filespec is required before the intended operation will initiate.
112	The maximum number of files has been opened. Subsequent attempts to open files without first closing a selected or open file will not be allowed.
113	The indicated source device and filetype are not in agreement. Re-check the fdp specification.
114	The indicated destination device and filetype are not in agreement. Re-check the fdp specification.
115	Device timeout. Received acknowledgement, but no reply.
116	An attempt has been made to access a protected system file.
117	Illegal filetype. Filetype number out of range.
118	Device number out of range.
119	A filetype (f)dp must be indicated in the command syntax.

Figure A.1 Processor Error Codes (continued)

ERROR #	DESCRIPTION
120	A device f(d)p must be indicated in the command syntax.
121	A port number fd(p) must be specified in the command syntax.
122	The communication routing must be defined in *the filespec syntax.*
123	The selected group of data must be defined with a filename.
124	The offset defining the starting point for data retrieval within the file is not valid. Check value and format for out of range, non-data.
125	The count specified is invalid. Check if needed, out of range, non-data.
126	The device indicated in the source and destination filespec cannot be identical.
127	A duplicate label already exists.
128	File mismatch. Compare command.
129	The register address attempted is not valid. Address above 8192 or below 0.
130	The filename indicated is invalid.
131	Write data error.
132	Unused
133	Insufficient data For write.
134	Unused
135	Invalid Read parameter.
136	A communication parity error exists.
137	Busy data transfer or weld in progress.
138	Communication overrun error
139	Communication framing error
140	Dropped unexpected reply message. Reply message received after device timeout (error 115), or no command expecting a reply.
141	Job space full. Maximum of eight jobs.
145	Write sequence not initiated.
146	Unused
147	Memory full.
148	Unused
149	No data found.
150	Cannot stop attached batch job.
151	Illegal communication to COMM Port.
153	*Error noted during a transfer.*

Figure A.1 Processor Error Codes (continued)

A.2.2 TAPE ERRORS

These errors are generated when performing tape operations and are displayed on the programmer as a "PROCESSOR ERROR". Refer to the table in Figure A.2 for a description of these errors.

ERROR #	DESCRIPTION OF TAPE ERROR
31	The end of the tape was encountered before the operation could be completed.
33	Tape data error detected - the block of data involved with the READ or WRITE operation was defective.
35	The tape cartridge is missing or is not seated properly, or the erasure prevention tab on the cartridge is set to the "write inhibit" position.
37	An attempt to skip or read a file was made when already past the last file on the tape.
53	Illegal tape format - an "erase track" is required.
55	The tape operation has been aborted. Retry the operation.

Figure A.2 Tape Error Codes

A.2.3 TRANSMISSION ERRORS

Errors due to problems with communication hardware (especially cables) are displayed on the programmer's screen as the message "TRANSMISSION ERROR", followed by a number.

The error codes that *most frequently* appear are the numbers 15, 17 and 74, and typically result from improperly fastened cables or baud rate mismatches. When an error with one of these three codes appears, check all cable connections and use the CLEAR/DEL key to clear the error indication. When all connections are secure, retry the operation. If the error persists, halting and cycling power to the processor causes the PRGMR port to revert to default baud and word structure and may allow successful communication with the programmer.

A.3 Programmable Controller System Operational Errors

These codes indicate errors somewhere in the overall programmable controller system and consist of a five-digit integer. These error codes are the contents of the data field (bits 1-16) of Control Register 8175. Seven possible classifications of operational error codes exist and are listed in the table in Figure A.3. Each of these classifications is explained later in this section.

NOTE: *The higher the error number, the higher the priority that the error has. For example, a "Slot Register Error" of any type will override any type of "Read After Write" error.*

OPERATIONAL ERROR #	DESCRIPTION
00001-00999	General error
01000-09999	Processor communication port error
10000-18192	Read after write error
19000-19016	Slot error
20000-28192	Slot register error
29000-29999	Miscellaneous error
30000-32700	CPU/LI error

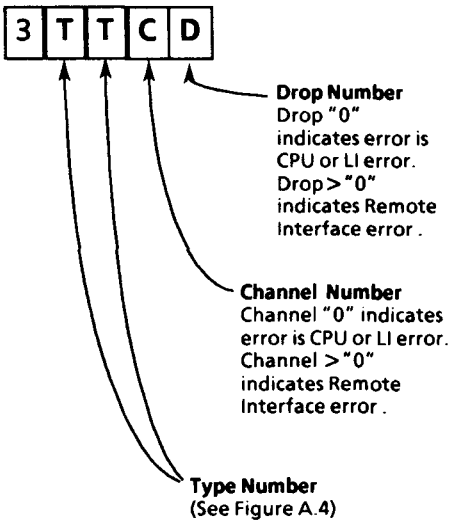
Figure A.3 System Operational Error Categories

When an error condition occurs, the appropriate error code is loaded into Control Register 8175 only if the register contains either zero or an error code of lower priority (a smaller number). The previous error code(s), if any, can then be stored in a 16-register block that is pointed to by pointer register 8107 (see Section 13.2).

The same operational errors are used by the Local/Remote Interface systems to indicate errors in devices under their control. Register 8175 or the error code in the programmer's status display indicates the problem slot of the Local Interface. Once located, inspect the error control register inside the Local Interface to determine which remote device contains the error.

A.3.1 CPU/LI ERRORS (30000-32700)

This error indicates a problem with either the processor (CPU) or Local Interface Module. Each of the five digits in this code represents a certain parameter as shown below:



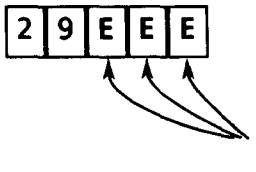
Refer to the table in Figure A.4 for the possible two-digit type numbers ("TT").

ERROR TYPE # (TT)	DESCRIPTION
00	Status register–READ/WRITE parity error.
01	Image RAM–READ/WRITE parity error.
02	Internal register data error.
03	Illegal PROM format.
04	Illegal opcode encountered while Model 650 in RUN mode.
05	Bus signal error (Interrupt detected on bus)
06	Software watchdog error.
07	Not used.
08	Software diagnostic error.
09	User memory–READ/WRITE parity error.
10	Illegal data in user memory.
11	PROM memory corrupted.
12	Illegal addressing map code.
13	Transmission error exceeded tolerance.
14	Loss of transmission.
15	Number of external I/O registers assigned exceeds the module's capacity.
16	Too many registers assigned to the drop.
17	More than 8 drops per channel are assigned.
18	Number of channels addressed exceeds module capacity, or channel 2 of LTI system is not rack-addressed.
19	Addressing map checksum error.
20	Bus error - watchdog timeout.
21	BPU diagnostic error.
22	Parity circuit non-functional.
23	Clear line error.
24	Executive scratch RAM error.
25	Watchdog tolerance error.
26	Hardware diagnostic error.
27	Module is inactive.

Figure A.4 CPU/LI (30000 - 32700) Error Codes

A.3.2 MISCELLANEOUS ERRORS (29000-29999)

The following illustration and Table A.5 identifies the miscellaneous error codes that may appear in the Model 650 processor.



ERROR CODE# (EEE)	DESCRIPTION
000	Subroutine nesting error*.
100	The time to process the timed interrupt subroutine has exceeded the time base*.
101	Timed interrupt routine is missing.
102	The time required to solve a rung has exceeded the tolerance set within a timed interrupt*.
103	Attempted to set up timed interrupt while interrupt is already running.
104	Attempted to return from a timed interrupt that wasn't running.
300	Mixed Primary/Back-up LTIs in CPU rack**.
301	*LTI timeout on EOS transfer****.
302	LTI timeout on start-up transfer***.
500	Register assignment conflict - number of assigned Local Interface registers must be at least 8 for one defined channel and 12 for two defined channels.
501	Insufficient number of local interface control registers.
510	RTI wired improperly-Ports A & B are mixed.
511	LTIs in CPU racks A and B have equal states of operation and no primary determination can be made in HALT.
512	Backup transfer system cannot be in RUN when primary transfer system is in DISABLE OUTPUTS.
513	Rack addressing between LTIs in racks A & B are not the same.
514	Backup cannot be run with test bit set when primary goes to HALT. Backup must also go to HALT.
515	Backup cannot become primary until synchronization has occurred and primary goes to HALT.
516	Backup unable to become primary because a remote bus error exists. This error is also produced by the primary to allow transfer to backup when primary is keyswitched to HALT.
517	Backup has lost synchronization and does not have RTC Failure Override bit set.
518	Two primary processors were found in RUN; this processor was halted with a bus error.

* *Control register 8184 contains the rung number where the error occurred. If 8184 is zero, check the rack addressing.*

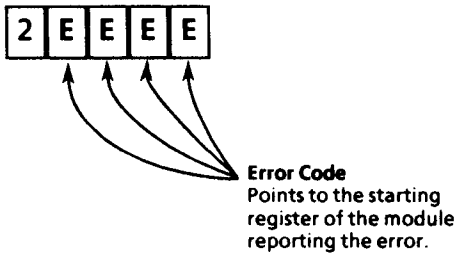
** *The data field of Control Register 8184 contains the bit table of primary or backup status (bit set means primary), while the status field of control register 8175 identifies the slots that contain LTIs (bit set means LTI is present).*

*** *The data field of Control Register 8184 identifies the first register assigned to the LTI that caused the error. The status field of 8175 contains the elapsed time (in milliseconds) that the scan was held.*

Figure A.5 Miscellaneous (29000-29999) Errors Codes

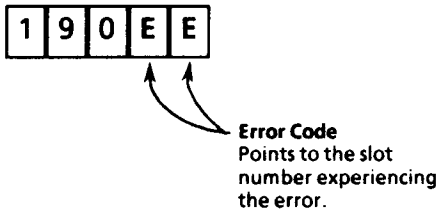
A.3.3 SLOT REGISTER ERROR (20000-28192)

In the event of a *slot register error*, the error code will consist of the *last four digits* of the five-digit error number as shown below. This four-digit number points out the first register number assigned to the slot containing a faulty register module. Further inspection of that register's status field will provide additional diagnostics.



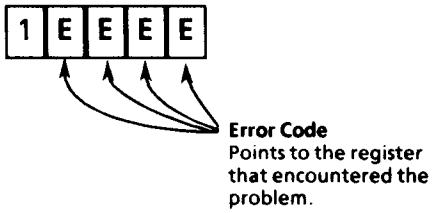
A.3.4 SLOT ERROR (19000-19016)

In the event of a *slot error*, the error code will consist of the last two digits in the five-digit error number, and will point out the slot number experiencing the error:

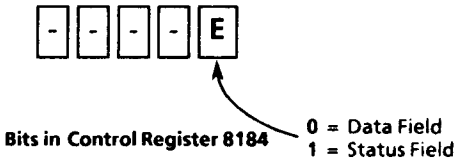


A.3.5 REGISTER READ AFTER WRITE ERROR (10000-18192)

A *read after write error* indicates a particular bit in a register has not maintained the condition written to it by the Model 650. The error code consists of the last four digits of the five-digit error number, and will point out which register encountered the problem.

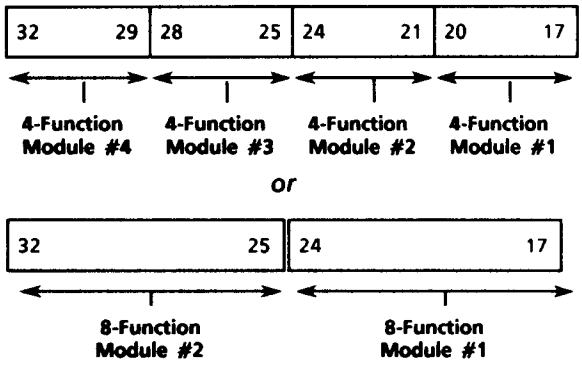


By displaying the status field (bits 17-32) of the control register containing the read after write error code, the bit or bits causing the malfunction will be indicated by a "1". Since each register contains both a status field and a data field, control register 8184 is used to indicate which set of bits contains the problem.



If the faulty register is used to control outputs, and Control Register 8184 indicates the bad bits are contained in the *data* field, the most likely cause for the problem is an output module.

The following diagram can help isolate the output module having a read after write problem. The status field of register 8175 is divided either four ways (for 4-function output modules), or two ways (for 8-function modules):



STATUS FIELD (BITS 17-32) OF REGISTER 8175

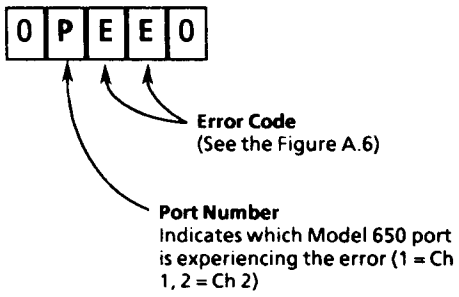
If any of the bits in the status field of register 8175 are set to "1", that bit was read incorrectly. If, however, *all* bits in the register are at "0", then a parity error was detected. At least one bit should normally be "1".

A.3.6 PROCESSOR COMMUNICATION PORT ERROR (01000-09999)

A communication port error occurs whenever the Model 650 does not receive a valid reply to a message it sends out of one of its serial ports. The error is cleared if a subsequent transmission *does* receive a valid reply.

To *clear* a communication port error, toggle the Model 650's keyswitch from HALT to RUN, or clear register 8175. Also, check remote device and cable connections.

The third and fourth digits in the five-digit error number identify what type of communication error occurred, while the second digit in the error number is the port in which the error occurred. The following diagram and table in Figure A.6 identifies the communication port errors.



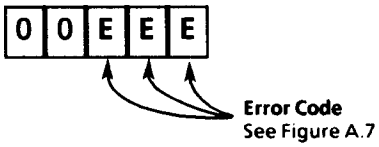
ERROR CODE #	DESCRIPTION
11	Communications overflow
12	Buffer overflow
13	Illegal data
14	Wrong reply (odd/even)
15	Checksum error
16	Framing error
17	Parity error
18	Unable to communicate through indicated port
19	Retry timeout

Figure A.6 Communication Errors

A.3.7 GENERAL ERRORS (00001-00999)

The error code for general errors consists of the last three digits in the five-digit error number as seen in the status display. Refer to the table in Figure A.7 for a description of the general error codes.

NOTE: *91X or 929 error codes indicate an Ethernet problem; refer to Section 15.7 for more information.*



ERROR CODE (EEE)	DESCRIPTION
900	Control registers corrupted *.
901	Data registers corrupted *.
902	Undervoltage Lockout is engaged (see Section 3.5).

Figure A.7 General Errors

ERROR CODE (EEE)	DESCRIPTION
960	HALT or HALT/RUN bit is set.
962	Tolerance is equal to or greater than the rate.
963	Timed interrupt timebase is less than 3.
964	GOTO and GOSUB are assigned same MARK**.
965	Illegal internal error code.
966	User subroutine stack error**.
967	A MARK rung with the same number is required**.
968	GOTO to MARK doesn't require a RTN rung**.
969	A subroutine MARK is lacking a RTN (return).
970	Scan time of Model 650 is greater than 1 second, or scan limit in register 8167 has been exceeded***.
971	Long scan due to LTI start-up transfer (check register 8184 for elapsed time in milliseconds).
979	Rack addressing is missing.
980	Keyswitch in RUN, but no program in memory.
983	Safeguarding rung is illegally programmed or missing.
984	Undefined register is programmed in user memory ***.
985	Processor is not installed in slot 1 though rack addressing is present.

* *Toggling the keyswitch allows the CPU to RUN. Control registers are normalized and data registers are cleared.*
** *S8184 will contain the MARK number.*
*** *S8184 will contain the rung number being executed when the error occurred.*

Figure A.7 General Errors (contiued)

A.4 Using Error Codes To Isolate Programmable Control System Faults

A.4.1 GENERAL CONCEPTS

When a system malfunctions, one of three situations can arise:

1. The whole system shuts down.
2. One or more drops shut down.
3. A run-time communication port error is detected.

Once the malfunction is observable as one of these three situations, the *system operational error codes* can be used to isolate the fault.

Recognition of the first category (entire system shutdown) is obvious; however, the second category (individual drop shutdown) can exist only when the system incorporates remote I/O and the override bit is set in the Local Interface Module (Refer to LI/RI Interface Instruction Bulletin 30598-175-01). The third category of malfunction (run-time COMM errors) deals with communications to external devices and requires that control register 8175 be monitored by the program for errors in the range 01000 to 09999. These types of errors are different from those described in the "Processor Errors" section in that they are *generated by the processor itself* and not by the external device. Each of the three malfunction situations is explained in the following sections.

A.4.2 WHAT TO DO IF THE PROGRAMMABLE CONTROLLER SYSTEM SHUTS DOWN

When the Model 650 halts, register 8175 should be the first place to check. The error code indicated will be within one of the following number ranges:

30000-32700 Processor/LI Error
29000-29999 Misc. Error
20000-28192 Slot Register Error
10000-18192 Read After Write Error
00001-00999 General Errors

If a *slot register error* (code **20000-28192**) is indicated, one of two courses of action can be taken depending on the specific register number indicated by the last four digits of the error code.

1. If the indicated register is the first assigned to a *register module*, interrogating the status field of that register will indicate the error. Refer to the appropriate register module Instruction Bulletin.

2. If the indicated register is the first assigned to a *Local Interface* Module (when using remote I/O), further interrogation of the LI error code register is required. Refer to Appendix D of the LI/RI Instruction Bulletin #30598-247-xx to determine the address of the LI error code register(s). If only a single LI exists in the system, register 8163 is the LI error code register.

   When the LI error code register is interrogated, the number will be in the following ranges:

   30000-32728 Processor/LI Error
   20000-28192 Slot Register Error
   10000-18192 Read After Write Error

In the case of a *read after write error* (code numbers 10000-18192), the affected bit or bits are indicated in the status field of the LI error code register.

The following checklist illustrates the procedure for examining a Programmable Controller system if it shuts down completely.

STEP 1: INTERROGATE CONTENTS OF REGISTER 8175:

- IF 30000-32700, Refer to Processor Error"
- IF 29000-29999, Refer to "Misc. Error"
- IF 20000-28192, Refer to STEP 2, following
- IF 10000-18192, Look at status field of S8175 for bits and refer to "Read After Write Error"

- IF 0001-9999, Refer to "General Errors", and interrogate control register S8184 when indicated.

STEP 2: DETERMINE THE SLOT TO WHICH THE REGISTER IS ASSIGNED USING THE ERROR CODE (REGISTER NUMBER) INDICATED BY THE LAST FOUR DIGITS OF THE ERROR NUMBER.

- If the indicated register is assigned to a *register module*, examine the status field of the register number indicated by the error code. Refer to the appropriate register module Instruction Bulletin for the meaning of the error number.

- If the indicated register is assigned to a *local interface* register, interrogate the LI error code and then refer to STEP 3, below.

STEP 3: INTERROGATE THE LI ERROR CODE REGISTER

- If the LI error code is 30000-32728, refer to *Processor* Error

- If LI code is 20000-28192, use the register number shown by the error code (last four digits). Determine the slot to which the register is assigned. The register module in that slot is the module that generated the error. Look at the status field of that register, then refer to appropriate register module Instruction Bulletin for the meaning of the error number.

- If the LI code is 10000-18192, look at the status field of the LI error code register for an indication of the affected bits and then refer to "Read After Write Errors."

STEP 4: MAKE SURE THAT THE LI MODULE IS SEATED PROPERLY

A.5 Example Malfunctions

Shown in Figure A.8, is an arbitrary programmable controller system experiencing one of four different malfunctions. Each area of malfunction is indicated by a circled number. The steps needed to isolate each type of problem, based on the previous checklist, are also explained.
The problem areas are :

1. The *Model 650 Itself*
2. The Local Register Module
3. The Remote Register Module
4. The Remote I/O Module
5. The Interconnecting Cables

A.5.1 EXAMPLE PROGRAMMABLE CONTROLLER SYSTEM SHUTDOWNS

The following example "breakdowns" are not intended to show all possible Programmable Controller system malfunctions, but attempts to show some of the more typical problems encountered with programmable controller systems.

Refer to Figure A.8 and to the checklists on the *previous pages when reviewing the following* examples.

NOTE: *Arbitrary register assignments have been added to this diagram to aid in the following explanation.*

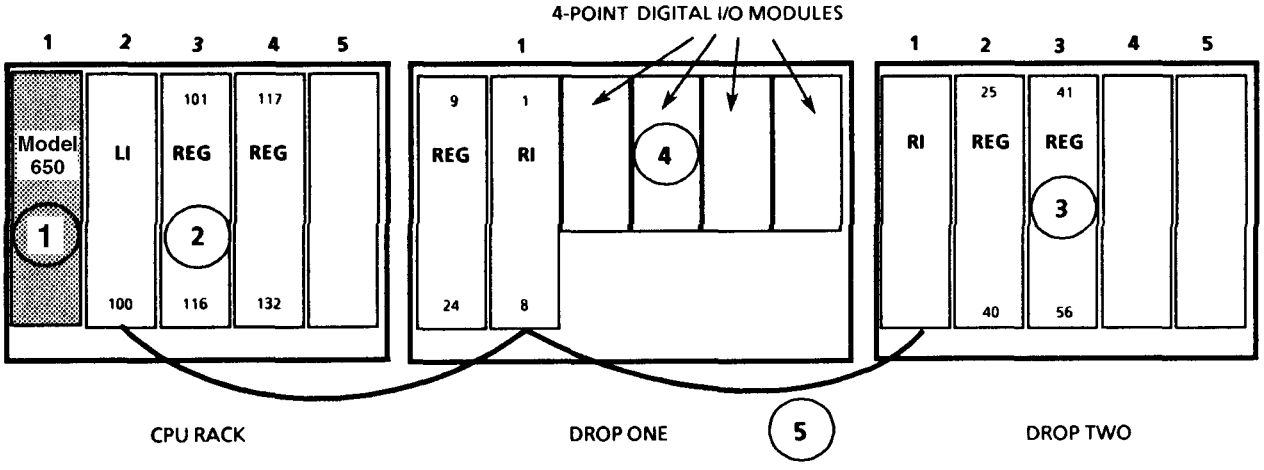


Figure A.8 Typical Programmable Controller System

(1) MALFUNCTION IN MODEL 650

A malfunction in the Model 650 itself would cause the system to shut down; the Programmable Controller system checklist on page A-12 should be used when working with this example.

Result of STEP 1: Since the malfunction occurred in the Model 650 itself, interrogating control register 8175 will result in an error code number which will help further identify the processor fault. A typical error code in this instance would range from 30000 to 32700.

(2) MALFUNCTION IN A LOCAL REGISTER MODULE (LOCATED IN CPU RACK)

A major malfunction in a register module located in the CPU rack would cause the system to shut down, regardless of whether any failure override bits are set. The Programmable Controller system checklist on page A-12 should therefore be used.

Result of STEP 1: Interrogating control register 8175 would result in a 20101 error code.

Result of STEP 2: The error code indicates the register module existing in slot #3 of the CPU rack has malfunctioned, since it is the module containing register 101 as the first register address. By looking at the status field of register 101 and then referring to the appropriate register module Instruction Bulletin, the specific error can be determined.

(3) MALFUNCTION IN A REMOTE REGISTER MODULE

A malfunction in a register module located in a remote rack will cause the system to shut down if the failure override bit in the LI module is not set. If this is the case, the "REMOTE RACK SHUTDOWN" checklist in Section A.5.2 should be used.

Result of STEP 1: Interrogating control register 8175 would result in a 20001 error code.

Result of STEP 2: The register number indicated in the error code is 001, which is the first register in the LI. Interrogating the LI error code register (in this case 8163) will provide additional information as to which module in the remote I/O system caused the shut down.

Result of STEP 3: Interrogating the LI error code register would result in a 20041. Since the register module in drop #2, slot #3 contains register 41, that module is responsible for the system shut down. By looking at the status field of register 41 and then referring to the appropriate register module Instruction Bulletin, the specific error can be determined.

4 MALFUNCTION IN A REMOTE DIGITAL I/O MODULE

A malfunction in a digital output module located in a remote rack will cause the system to shut down if the failure override bit in the LI module is not set. If this is the case, the "REMOTE RACK SHUTDOWN" checklist in Section A.5.2 should be used.

Result of STEP 1: Interrogating control register 8175 would result in a 20001 error code.

Result of STEP 2: The register number indicated in the error code is 001, which is the first register in the LI. Interrogating the LI error code register (in this case 8163) will provide additional information as to which module in the remote I/O system caused the shut down.

Result of STEP 3: Interrogating the LI error code register would result in a 10001. Looking at the status field of 8163 will provide the bit mask that indicates which output module associated with register 001 caused the failure.

5 COMMUNICATION ERROR BETWEEN LI AND RI MODULES

A communication error that occurs between local and remote interface modules will cause the system to shut down if the failure override bit in the LI module is not set. If this is the case, the "REMOTE RACK SHUTDOWN" checklist in Section A.5.2 should be used.

Result of STEP 1: Interrogating control register 8175 would result in a 20001 error code.

Result of STEP 2: The register number indicated in the error code is 001, which is the first register in the LI. Interrogating the LI error code register (in this case, 8163) provides additional information as to which module in the remote I/O system caused the shutdown.

Result of STEP 3: Interrogating the LI error code register would result in a 31412. Referring to section A.3.1 reveals that a transmission error has occurred on channel 1, drop 2. Transmission errors result from many causes, including defective cable, electrical noise, LI or RI communication chip malfunction, inadequate power, improperly set DIP switches, etc. Further investigation is required to pinpoint the problem.

Setting the failure override bit for the respective channel experiencing the failure (bit 8162-13 for this example) allows the processor to ignore the error and remain in RUN, although information associated with the failed drop is no longer valid. Similarly, setting the auto restart bit for the channel (bit 8162-15) causes the LI module to keep attempting to reestablish communication. Thus, in the event the transmission error was caused by a transient condition, the failed drop experiences only a momentary disruption.

A.5.2 REMOTE RACK SHUTDOWN

When a remote rack(s) halts without causing the rest of the system to halt (Failure Override bit set for that channel), the error code register of the Local Interface module controlling that drop will contain the error number of the fault. When the LI error code register is interrogated, the number will be in the following ranges:

30000-32700	Model 650/LI Error
20000-28192	Slot Register Error
10000-18192	Read After Write Error

In the case of the 10000-18192 Read After Write Errors, the affected bits are indicated in the LI error code register.

The following checklist illustrates the "REMOTE RACK SHUTDOWN" scenario:

STEP 1: INTERROGATE CONTENTS OF APPROPRIATE LOCAL INTERFACE ERROR CODE REGISTER

If **30000-32728**: Refer to Processor Errors

If **20000-28192**: Using the register number from the error code (last four digits), determine the slot to which the register is assigned. *The register module in that slot generated the error.* Look at status field of that register, then refer to the appropriate register module Instruction Bulletin for the meaning of the error number.

If **10000-18192**: Look at the status field of the LI error code register for an indication of the affected bits. Refer to "Read After Write Errors."

Using malfunctions #3 and #4 in Figure A.1 as examples, and the above checklist, go through the following steps to isolate each type of problem.

The two main problem areas are the (1) remote *register module* and (2) the remote *I/O module*

1. A malfunction in a *register module* located in a remote rack will cause only that rack (drop) to shut down, assuming the failure override bit in the LI module is set. If this is the case, the "REMOTE RACK SHUTDOWN" checklist should be used.

Following are results of applying the "remote rack shuts down" checklist:

Result of STEP 1: Interrogating the LI error code register (8163 in this case) would result in a 20041. Since the register module in Drop #2, Slot #3 contains register 41, that module is responsible for the system shut down. By looking at the status field of register 41 and then referring to the appropriate register module Instruction Bulletin, the specific error can be determined.

A malfunction in a *digital output module* located in a remote rack will cause only that rack (drop) to shut down, assuming the override bit in the LI module is set. If this is the case, the "REMOTE RACK SHUTDOWN" checklist should be used.

Result of STEP 1: Interrogating the LI error code register (8163 in this case) would result in a 10001. Looking at the status field of 8163 will provide the bit mask indicating which output module associated with register 001 caused the failure.

A.5.3 "PRGMR" OR "COMM" PORT ERRORS

These errors are different from the processor errors associated with communications listed earlier in this Appendix in that processor communication port errors are detected by the Model 650 rather than the external communicating device.

The only processor error that has any relationship to processor communication port errors (01000-09999) is processor error code 13. When error 13 is indicated in a status register of a communication rung, interrogation of control register 8175 will result in the display of one of the processor communication port errors (01000-09999).

Communication port errors can also be detected when an external device is asking for information from the Model 650 or other local processor.

NOTE: *In these examples, there is no communication rung status register to indicate a fault. It is up to the user to program monitoring and annunciation of these types of errors.*

APPENDIX B
OPERATING CONSIDERATIONS WHEN THE MODEL 650 IS USED WITH THE LOCAL/REMOTE TRANSFER INTERFACE MODULES

The purpose of this appendix is to point out some of the operating characteristics to be expected when using the Local/Remote Transfer Interface (LTI/RTI) modules (Class 8030 Type CRM-230/232 modules) with the Model 650 processor.

NOTE: *FAMILIARITY WITH THE LTI/RTI INSTRUCTION BULLETIN (BULLETIN #30598-251-XX) IS REQUIRED WHEN USING THIS APPENDIX.*

Because the LTI/RTI Instruction Bulletin (#30598-251-XX) is geared to the Model 500 and 700 processors and was written prior to the Model 650's creation, some of the information contained in that bulletin DOES NOT APPLY TO Model 650 OPERATION.

WARNING

Certain application considerations must be observed when using the Model 650 in systems with Transfer Interfaces. THIS IS PARTICULARLY TRUE IN SYSTEMS ORIGINALLY DESIGNED FOR MODEL 500 OR MODEL 700 PROCESSORS. Until this section has been read and understood, DO NOT use Model 650s in LTI/RTI systems. Equipment damage or bodily injury may result if this section is not followed in detail.

B.1 General Discussion

As described in the Local/Remote Transfer Interface System Instruction Bulletin (30598-251-XX), system redundancy is actually accomplished via the LTI modules, *not* the processor. Through the LTI-to-LTI high-speed Register Transfer Channel (RTC) link and the respective LTI-to-RTI I/O channel links, the primary and backup LTI modules exchange the information necessary to maintain the backup in a state of readiness to seize control of the system's I/O in the event of a failure in the primary rack.

Because the register data actually resides in the image table of the Model 650, it must be passed from the primary Model 650 CPU to the primary LTIs, then to the backup LTIs, and finally to the backup Model 650 CPU. The quantity of registers to update is a function of rack addressing and the *type* of transfer (*startup* or *end-of-scan*) that is occurring.

B.2 Rack Addressing

Since the Model 650 occupies a single slot, the first LTI module can be installed in slot 2 of the CPU rack. When rack addressing the system, *assign only as many registers to the LTI slot as is required by the user program.* Because registers assigned to the LTI(s) are exchanged during a startup transfer, any registers that are unnecessarily assigned will extend the startup transfer time. Thus, when laying out the system, *use storage registers with low-numbered addresses.*

Startup transfer time can be further reduced by setting the Startup Transfer Register equal to the number of registers that are assigned to the LTI slot (the Startup Transfer Register is the first of four registers assigned to LTI channel 2, drop 1). As explained in Section B.3 below and in Section 5.4.2 of the LTI/RTI bulletin, this practice minimizes the number of registers that the primary LTI writes to the backup LTI.

In a *multiple LTI* system, group the data storage registers in the *last* (rightmost) LTI, as is shown in Section 5.3.6 of the LTI/RTI bulletin. As previously mentioned, assign only as many registers to the LTI slot as is required by the user program. Also, the Startup Transfer Register for each LTI should be set equal to the last (highest) register assigned to the LTI slot.

B.3 Startup Transfer Delay

Use the following equation to estimate total startup transfer delay time:

DELAY (msec) = (0.03 × RA) + (0.26 × ERT) + (0.17 × IRT)

Where:
RA is the total number of registers *RACK* ADDRESSED to the LTI, **ERT** is the number of EXTERNAL I/O REGISTERS in the LTI being transferred, and **IRT** is the number of internal storage registers in the LTI being transferred.

Assume, from the example of the LTI/RTI Instruction Bulletin Section 5.4.2, that 1,000 registers are to be transferred from the primary to the backup (as specified by setting S73 = 1,000). In accordance with Section B.2 and the above equation, if 1,000 registers are rack addressed to the LTI, total startup transfer delay is:

(0.03 × 1000) + (0.26 × 76) + (0.17 × 924) = 207 milliseconds.

If the startup transfer was *not* preset to limit (to 1000) the number of registers transferred and the maximum number of registers were rack addressed to the LTI (4096), the startup delay would be much longer. The startup delay now becomes:

(0.03 × 4096) + (0.26 × 76) + (0.17 × 4,020) = 826 milliseconds

In a multiple LTI system, startup transfer delay time will be longer than indicated by the above equation. In many cases, the delay time will exceed one second, and will be based on system configuration factors such as the number of LTIs present and the register distribution between them. When this occurs, error 971 will appear in register 8175 and the actual elapsed time in milliseconds will be posted in register 8184. It may be possible to decrease this time by following the recommendations of Section B.2, especially rack addressing to the LTIs only the quantity of registers utilized in the user program.

NOTE: *The processor's scan time limit (default is one second) is not active during a startup transfer. If the scan limit is exceeded, error 971 is posted in the backup processor as a flag to note the long transfer time, and is not indicative of a system error or improper operation.*

B.4 End-Of-Scan (EOS) Transfer

At the end of each ladder scan, 32 registers can be transferred from the primary LTI to the backup LTI via the Register Transfer Channel (RTC). Transferring these registers is intended to maintain synchronization of critical internal registers that might otherwise differ. This difference may be due to communication with external devices, timer drift, etc.

NOTE: *Unlike the SY/MAX Models 500 or 700, the position of the Model 650's 32-register EOS transfer block is FIXED. IT MUST OCCUPY RELATIVE REGISTERS 260-291 ASSIGNED TO AN LTI. Thus, any program written for the Model 500 or 700 that uses the EOS Transfer feature must be configured for this restriction.*

To set up the EOS transfer, preset the contents of the EOS Transfer Register to the address of the 260th register assigned to the LTI (the EOS Transfer Register is the second of four registers assigned to LTI channel 2 drop 1). To execute the transfer, set the EOS Transfer Register bit (bit 16 of the backup LTI's RTC Control Register). This causes the values of the 32 registers (registers 260 through 291) to be transferred at the end of every scan.

NOTE: *Enabling the EOS Transfer feature adds approximately six milliseconds to the Model 650's scan time.*

Refer to the multiple LTI example in Section 5.3.6 of the LTI/RTI bulletin. Since the first two LTIs do not have at least 291 registers assigned to them, no EOS transfer is possible. The rightmost LTI, however, could transfer registers 396-427 (relative registers 260-291). To activate the transfer, preset the EOS Transfer Register (S202) equal to 396, and set the EOS Transfer Register bit to "1" (bit 16 of register 8155).

NOTE: *An easy way to determine the starting address for the EOS transfer block is to add 259 to the first register assigned to the LTI slot (provided, of course, that at least 291 registers are assigned to the LTI).*

If more than 32 registers need to be transferred, use a matrix instruction within the application program to transfer different 32-register blocks into the EOS Transfer block (registers 396-427 in the above example). Refer to Section B.8 for an application example.

B.5 I/O Update

As described in Section 14.2 of this bulletin, the Model 650 performs an I/O update of modules in the CPU rack at the end of every scan. For an LTI, *the first 291 registers (if this many are assigned) MUST be transferred between the image table of the Model 650 and the LTI(s). This transfer, along with some* "overhead" due to LTI-to-processor handshaking, results in an I/O update time of about 15 milliseconds per LTI. This 15 milliseconds must be added to the scan speed equation of Section 14.2.2 of this bulletin.

NOTE: *The "ER" (unfragmented register) and "FER" (fragmented register) terms in the scan speed equation apply to NON-LTI modules in the CPU rack, and will therefore be zero or so close to zero that they can typically be disregarded.*

The I/O update time of 15 milliseconds per LTI assumes that the assigned registers are *unfragmented*; LTI update times will increase by 0.45 milliseconds for every *fragmented* register (a register containing both input and output points).

NOTE: *In a multiple LTI system, the I/O updates are additive. If fewer than 291 registers are assigned to a given LTI the I/O update time for that LTI is proportionally less than 15 milliseconds.*

B.6 Forcing

WARNING

The forced state of I/O DOES NOT TRANSFER when system control is switched from Primary to Backup. During a transfer, BE AWARE OF ANY CHANGES IN SYSTEM BEHAVIOR THAT MAY OCCUR DUE TO ACTIVE FORCING. Incorrectly forced I/O may cause personal injury and equipment damage.

Forcing is implemented differently in the Model 650 than in Model 500 or 700 processors. As described in Section 14.4 of this manual, forcing in a Model 650 is a condition that should be applied to *external I/O only*. Forcing bits of an *internal* storage register causes a programming device to display the *forced* state, while the image table in the Model 650 actually processes the information based on the *logic* state.

NOTE: *If a bit of an internal storage register is forced in a primary, a Startup Transfer will cause the same bit in the backup to reflect the forced state. This is a "one-shot" transfer that may cause unintended results, especially if the bit(s) are within a data storage register not controlled by the user ladder program. Avoid this situation by following Section 14.4's recommendations to perform forcing ONLY ON EXTERNAL INPUTS AND OUTPUTS.*

B.7 Primary and Backup READ/WRITE Bit Exchange

Section 5.4.5 of the LTI/RTI bulletin describes a method for exchanging the status of eight user-selectable bits residing in the fourth register that is assigned to the RTC (Channel 2, Drop 1). THE Model 650 DOES NOT SUPPORT THIS BIT-EXCHANGING CAPABILITY.

B.8 EOS Transfer Example

This example describes how to implement an end of scan (EOS) register block transfer between the primary and backup LTI's when the block to be transferred is outside the fixed register location. Because the block of 32 registers associated with this EOS in a Model 400, 600 or 650 is fixed as relative registers 260 through 291, additional registers outside this range must be multiplexed through them to the backup processor. The following provides an example of how to accomplish this multiplexing operation using GOTO/MARK loops with INCREMENTAL LET MATRIX ladder rungs.

The ladder rungs are divided into two groups:

> The primary group of rungs controls the uploading of the fixed block of transfer registers into the primary LTI from other locations in the primary processor.
>
> The secondary or backup group of rungs downloads the register data transferred through the fixed block area to their correct locations in the backup processor.

Since either processor can become primary or backup at any time, it is necessary to have both sets of ladder rungs resident in each processor. To determine which set of rungs are operated on in each processor, the primary/backup status bit 8161-20 from the LTI's Remote Transfer Channel (RTC) Status register is used in conjunction with a GOTO branch to allow the correct rungs to be scanned depending on that processor's current primary or back-up status.

The primary group of ladder rungs should always be placed at the end of the processor's mainline ladder program but before any MARK START SUBROUTINE area. The backup group of ladder rungs should always be placed ahead of the processor's mainline ladder program but after the LTI's configuration setup rungs. Placing these rungs in different areas of the processor's scan ensures that all ladder functions that could affect the primary's registers are completed before they are transferred to the backup, and then uploaded by the backup processor's scan prior to using these register values in the backup.

The following describes the rungs used to accomplish the register transfer multiplexing operation between the primary and backup processors.

Rack addressing for this example assumes that at least 1000 registers are assigned to the LTI slot, that registers 5 through 8 are assigned as the RTC registers (LTI Channel 2, Drop 1), and has registers S260 through S291 acting as the fixed 32 register transfer block or window between the primary and backup LTIs.

To manage the multiplexing operation, a pointer value indicating the location where the transfer block was uploaded from the primary processor must be sent to the backup processor so the registers can be downloaded to the correct location. The first register (S260) of the fixed transfer block of 32 is reserved for this duty. The remaining 31 registers of the fixed transfer block contain the actual multiplexed register data to be transferred.

The following RTC and Control registers are important to LTI operations:

REGISTER	DESCRIPTION
S5	Startup Transfer Register - determines the number of registers transferred from the primary processor to the backup processor at startup. A default value of zero will cause all 4096 registers of the LTI to be transferred.
S6	End-of-Scan Transfer Register - determines the starting register location of the block of 32 registers to be transferred from the primary to backup processors. In the Model 400, 600 and 650 processors, the relative starting location of this block of 32 end-of-scan transfer registers is fixed and is defined as the 260th register assigned to the LTI; the user must preset this information as part of the ladder program.
S7	Processor Scan Synchronization Register - determines the allowed variation between scan times of the Primary and Backup processors. Set to 0 for default value of 100 msec.
S8	*Primary/Backup Read/Write Bit* Exchange - not supported by the Model 400, 600, or 650 processors. Should be set to 0.

S8176 Processor Control Register - Bit 23 (Normal Program Scan) allows the ladder rung in which it is used to establish preset conditions prior to a processor commencing normal ladder scanning.

S8161 Remote Transfer Channel (RTC) Status Register - Bit# 20 of this status register determines whether the LTI being used is either the primary (Bit# 20 is ON or 1) or backup (Bit# 20 is OFF or 0).

Bit# 16 - *EOS Transfer: When set ON* (1) in a Backup LTI that is scanning ladder, the Backup LTI will request an End-of-Scan Transfer of the fixed 32 register transfer block from the primary.

The following internal processor registers were chosen as temporary storage locations for the multiplexing of data registers through the fixed 32 register block between the primary and backup processors. These registers should be chosen from an unused internal register area not involved in the end-of-scan block transfer or other storage purposes. The registers shown below were arbitrarily chosen *for this example only.*

REGISTER DESCRIPTION

S260 First register of the fixed transfer register block. *Used* as the pointer to identify the starting address of the block currently being multiplexed.

S292 Starting location of a block of data registers to be transferred from the primary processor to the backup processor at end-of-scan.

This starting location can be calculated by subtracting one (1) from the starting register address of the block of internal data registers to be transferred at end-of-scan. This starting location will be passed to the backup processor *through* the first relative fixed register (S260) of the transfer window.

This register is later used as a pointer in one of the multiplexing matrixes to keep track of the starting location for the next block of registers to be transferred through the fixed register window (S260-S291).

S293 This register is used as a pointer in one of the multiplexing matrixes to keep track of the next register to be loaded or unloaded. This register should be initialized to zero and cleared after each 32 register end-of-scan transfer is complete.

The status bit# 17 of the above matrix pointer registers is used in the example as a contact to exit the GOTO/MARK loops since it will come ON (1) whenever the incremental matrix PNTR register value equals its SIZE.

S294 This intermediate register is used in the matrix routines to pass the register data between the fixed transfer block register and the processor's storage location.

The following example can be broken up into three groups of rungs, each regulating an aspect of the End-of-Scan (EOS) register transfer. The first group, rungs 1 through 3, will set-up or initialize the RTC registers, EOS Transfer Bit and the first data register's starting location of the block of registers to be transferred. The second group, rungs 4 through 11, are the rungs required by the backup processor to *download or distribute* the data registers passed to it through the fixed transfer register block by the primary processor. These first two groups of rungs should always be placed ahead of the processor's operational ladder program to allow early *initialization of the transfer process and the* downloading of current data from the primary processor. The last of the three groups of rungs in this example, rungs 12 through 20, are used by the primary processor to upload the block of data registers for transfer to the backup processor through the fixed transfer block. This last group of rungs should always be placed at the end of the mainline ladder program, before the MARK START SUBROUTINE rung. This will allow the primary processor to upload current data register values to the backup processor.

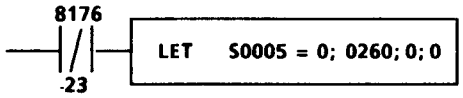
The following is an example of the ladder rungs used *to perform this EOS register block transfer.* The word "block" must be emphasized here since this routine will only operate on one continuous unbroken sequence or block of internal processor data registers. Note that these registers must be rack addressed to the LTI slot in order to be exchanged. In this example, data registers 501 through 1000 will be transferred from the primary to the backup Model 650 processor.

This 500-register transfer will require 17 processor ladder scans to complete, since only 31 registers can be transferred per each EOS interval. Because either processor in a transfer system can be primary or back-up, all twenty of the following rungs used in this multiplexing example should be included in both the primary and backup processor ladder programs. This will allow the status of either processor to change at any time and still maintain the EOS register transfer multiplexing process. Each ladder rung of this routine is preceded by a brief description of its purpose.

SETUP AND INITIALIZATION RUNGS (1 through 3)

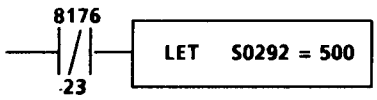
RUNG 0001

This set-up rung presets the RTC registers assigned to LTI channel 2, drop 1. The second argument's (S6) value of "260" matches the relative starting register address of the fixed register block window used for EOS transfers. Note S5 should normally be set equal to the largest register address assigned to the LTI slot; this example uses the default value of 0.



RUNG 0002

This rung loads register S292 with the starting location (501 - 1 = 500) of the first register of the block of 500 to be transferred by the primary processor after system startup.



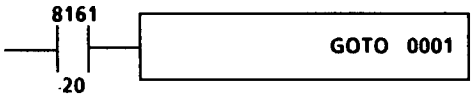
RUNG 0003

This rung sets the EOS Transfer Bit in the backup LTI only.

BACK-UP (REGISTER DOWNLOAD) RUNGS (4 through 11)

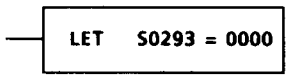
RUNG 0004

Branch around rungs 5 through 10 if this is the primary processor.



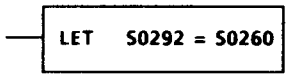
RUNG 0005

Initializes subsequent matrix rung's pointer (PNTR) register to zero.



RUNG 0006

Load data register S292 with the starting location where the following block of 31 registers transferred to the backup processor are to be located.

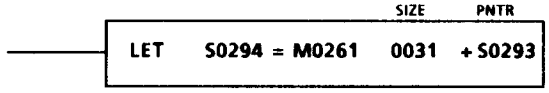


RUNG 0007

The start of the backup processor's download loop.

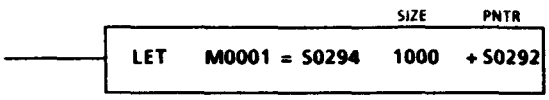
RUNG 0008

Incremental LET matrix rung that uploads each of the 31 fixed transfer block registers into register S294. PNTR register S293 increments for each pass through the loop (note--pointers used in "incremental matrix" statements pre-increment before execution of the rung). The size of this matrix rung is always 31 since the first register (S260) transfers the starting location of the block of 31 registers that follow to the backup processor.



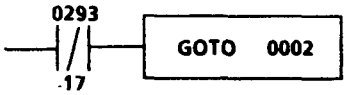
RUNG 0009

Incremental LET matrix rung that downloads each of the 31 registers transferred at the EOS to the correct location in the backup processor. The starting location for this storage operation was passed to pointer register S292 through transfer block register S260 in rung 6. The size of this matrix should always be equal to the last address of the total block of data registers to be transferred to the backup processor (in this case, register 1000).



RUNG 0010

Continue to loop until all 31 registers transferred at EOS to the backup processor have been downloaded to the correct locations before proceeding to the mainline ladder program.



RUNG 0011

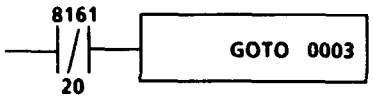
Last rung of the backup group, branched to from rung 4 if processor is primary.

This area (following the last rung of the backup group) should contain the processor's mainline ladder program. The following primary group of rungs should be placed at the end of the mainline ladder program but before any MARK START SUBROUTINE rung.

PRIMARY (REGISTER UPLOAD) RUNGS (12 through 20)

RUNG 0012

Branch around rungs 13 through 19 if this is the backup processor.



RUNG 0013

Initializes subsequent matrix rung's pointer (PNTR) register to zero.

RUNG 0014

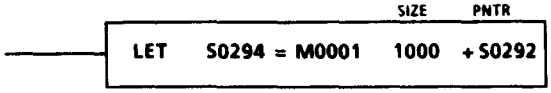
Load the first register (S260) of the fixed register transfer block with the starting location of the following block of 31 registers to be uploaded to the backup processor.

RUNG 0015

The start of the primary processor's upload loop.

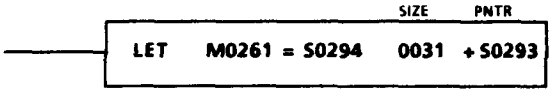
RUNG 0016

Incremental LET matrix rung that preloads each of the 31 registers to be transferred at the EOS from the locations in the primary processor. The starting block location for this routine was loaded into pointer register S292 by either rung 2 or 19. The size of this matrix should always be equal to the last address of the total block of data registers to be transferred to the backup processor (in this case, register 1000).



RUNG 0017

Incremental LET matrix rung that loads each of the 31 fixed transfer block registers through register S294. The size of this matrix rung is always 31; the maximum number of registers that can be transferred by this routine to the backup processor at EOS.

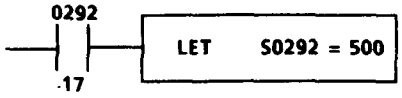


RUNG 0018

Branch to rung 15 and continue to loop until all 31 registers to be transferred to the backup processor at EOS have been loaded into the fixed register transfer block.

RUNG 0019

The pointer register S292 was incremented until equal to the last address (1000) of the register block transferred, and activated contact S292-17 to restart the process again. This rung then reloads pointer register S292 with the original starting location (501 - 1 = 500) of the entire register block.



RUNG 0020

Last rung of the primary group, branched to from rung 12 if processor is backup.

This example illustrates the ladder rungs required for multiplexing more than 32 registers (in this case, registers 501 through through 1000) through the LTI's EOS transfer block. The End-of-Scan transfer ladder rung routine shown in this example can be modified to fit any desired size register block to be transferred by changing the values of the starting location and last register address of the block in rungs 2, 9, 16 and 19. Further modification of this routine will be required if relative fixed register block transfer window addresses other than S260-S291 are used by the LTI's, as could happen in a multiple LTI system.

APPENDIX C SUPPLEMENTARY RACK ADDRESSING INFORMATION

This appendix provides additional information on register allocation and updating, and supports Sections 3.1, 3.6, 3.7 and 14.2.

C.1 Model 650 System Register Updating

The Model 650 contains 8000 on-board registers; this is the maximum number it is capable of addressing. Adding a Local Interface (LI) module *will not* increase the addressing capability beyond this number. Thus, the Model 650 already has all of the data storage registers it can handle. By the same token, any registers that have to be updated in other modules will add to the scan time.

Simply stated, the more registers the Model 650 needs to read and write on the bus, the longer the scan time will be, as indicated in the scan speed equation of Section 14.2.2.

To some extent, the Model 650 makes "decisions" to optimize throughput. For example, the Model 650 will interrogate register modules in the CPU rack which have had registers assigned to them to determine the module type. If the module is *not* a Local Interface, the Model 650 determines whether it *is an input or output device and then simply reads or* writes the assigned number of registers at the end of each scan. If the module interrogated *is* a Local Interface, some additional decision making will occur, as discussed below.

If no module is present, the assigned registers revert to the image table of the Model 650 and behave like regular storage registers.

C.2 Rack Addressing A New System

For a listing of ground rules to follow when rack addressing, refer to Section 3.6 for discussions on *"Determining Rack Addressing Needs"* and *"Rack Addressing Register Allocation"*.

C.2.1 GENERAL RULES

The Model 650 initially defaults to 8000 registers assigned to CPU slot #1; this applies to installations *in which the Model 650 is installed in a digital I/O* rack with no other register modules and no remote I/O.

Registers may or may not require assignment to CPU slot #1, depending on whether the processor is installed in a digital or register rack as discussed below.

When assigning registers, start with the CPU rack. If installed in a digital I/O rack, slot #1 needs to have sufficient registers assigned to handle the local digital I/O. If installed in a register rack, it is not *necessary to assign registers to slot #1. Each slot in* the CPU rack should have as many registers assigned to it as required to satisfy external I/O needs. Once registers have been allocated in the CPU rack, whatever registers remain unassigned (up to 8000) are available for data storage.

There is no need to assign registers in multiples of four. Also, it is now possible to force the 16-point and 32-point digital I/O when used in the CPU rack. If a slot contains a Local Interface, *assign only enough registers to service external I/O;* any data storage registers assigned to a Local Interface will adversely *affect throughput. However, be sure to anticipate* any future expansion.

Avoid "fragmenting" registers when setting up a programmable controller system (discussed in Section 14.2.3 of this manual). A fragmented register is one that contains both inputs and outputs, as could occur in systems which use four-point or eight-point digital I/O. For example, if bits 1 through 8 of register 1 are inputs, and bits 9 through 16 are outputs, then register 1 is fragmented. If, however, *all* bits of register 1 are inputs or outputs then no fragmentation exists.

NOTE: *For systems which use a PDD/PDR combination, refer to Section C.2.3.; for Local Transfer Interface considerations, see Appendix B.*

Fragmenting registers increases scan time. From the scan time equation in Section 14.2.2:

ST = [K × (LUP)] + [0.04 × (ER)] + [0.45 × (FER)] + COM

ST is the SCAN TIME in milliseconds.

K is the LADDER SCAN SPEED per K of user program.

LUP is the LENGTH OF USER PROGRAM in Kwords.

ER is the number of EXTERNAL REGISTERS.

FER is the number of FRAGMENTED EXTERNAL REGISTERS.

COM is the time requiered to service the communication ports, 5 msec maximum.

Note how scan time is extended an additional 450 microseconds for each fragmented register.

As with the Model 300 (which also uses an image table), the Model 650 uses bit 7 of primary control register 8176 to activate an immediate I/O update. *Unlike* the Model 300, the registers to be updated are now variable and can be user-specified. Refer to the description of bit 7 of register 8176, and control registers 8105 and 8106, in Section 13.2 of this manual.

MODEL 650 IN A DIGITAL RACK

If the Model 650 is used in a digital I/O rack which contains one or more register modules (as could occur with the HRK-150 rack), assign enough registers for all local digital I/O to the CPU slot (slot #1). Next, assign registers in ascending order to slot 2 (and to slots #3 and #4 if the rack used is an HRK-150).

MODEL 650 IN A REGISTER RACK

If the Model 650 is used in a register rack, *registers DO NOT need to be assigned to the Model 650 itself* (CPU slot 1); assign registers in ascending order beginning with slot #2.

C.2.2 LOCAL INTERFACE (LI) UPDATING

If the Model 650 detects that registers are assigned to a Local Interface module in a CPU rack slot, more decision-making occurs to optimize throughput.

When a Local Interface is detected, the Model 650 next checks to see if any remote drops are assigned. If not, *no* registers will be transferred from the image table of the Model 650 to the LI. In essence, the Model 650 makes a decision to not "waste" any processing time writing storage registers to an LI, since storage registers already exist in the Model 650 itself.

As soon as one or more remote drops are assigned, however, the Model 650 must update the LI registers that are used on the remote drops. These LI registers are now assigned to external I/O instead of simply data storage. Since an LI cannot exchange more than 255 registers assigned to remote drops, the Model 650, in an attempt to optimize throughput, never transfers more than 255 registers to an LI, *even if the user has assigned the additional registers to the LI slot.* This could occur if an existing rack addressing configuration is being used. This relates to a previously described principle for optimizing throughput: *only assign those registers to a LI necessary to service external I/O.*

EXAMPLE: If an LI has four drops assigned, with eight registers per drop, a total of 32 registers are required for external I/O. Use the scan time equation on the previous page to estimate the scan speed impact. Assuming no fragmented registers, if 32 registers are assigned to the LI, scan time is increased by **0.04 milliseconds x 32 = 1.3 milliseconds**.

If, however, the LI had *512* registers assigned to it, the scan time increases by **0.04 milliseconds x 255 = 10.2 milliseconds** for the same user configuration. Thus, rack addressing can have a substantial impact on system scan time.

Note how the Model 650 tries to compensate for poor register allocation: even though 512 registers were assigned to the LI, the Model 650 is intelligent enough to only update the first 255. This helps in situations where existing rack addressing configurations are used.

CRM-210/211/214

There is no reason to use a CRM-211 Local Interface Module in place of a CRM-210 in a Model 650 system, since the extra data storage registers are not used by the Model 650. Also, the CRM-214 (LAI) register allocation and updating methods are identical to Local Interface register updating.

C.2.3 PDD/PDR UPDATING

The Parallel Digital Driver/Receiver modules (Class 8030 Type EQ5138-G1 and -G2) were originally designed to allow real-time digital I/O bus updates with the Model 500 and Model 700, which *don't* have the Model 650's on-board image table. Since the Model 650 *does* use an image table, oscillating output behavior may be observed when the Model 650 is used with the PDD/PDR. This behavior is due to *register wraparound.*

Wraparound is discussed in the PDD/PDR Instruction Bulletin under "Application Considerations With the Model 300". It is a phenomenon in which registers that are *assigned* to the PDD, but *do not correspond* to physically existing external I/O, cause digital bus update problems. Wraparound occurs because the registers that do not correspond to physically existing external I/O interfere with the registers that do. Wraparound can be avoided by properly addressing the PDD/PDR.

For COMPLETE information on proper rack addressing, see Section 3.6.4 for a discussion on "Rack Addressing Register Allocation".

In brief, the guidelines for proper PDD/PDR rack addressing are:

1. **If a single remote rack is used,** it must be connected to channel 1 of the PDD. When rack addressing, only assign as many registers to the PDD as required by the external I/O of the rack (consistent with one of the previously established "general rules"). For example, if the remote rack is a CRK-300, DRK-300, or HRK-100, assign 4 registers to the PDD; if the remote rack is an HRK-200, assign 8 registers to the PDD.

2. **If two remote racks are used,** the rack connected to channel 1 of the PDD *must* be an **HRK-200**. When rack addressing, again only assign as many registers to the PDD as required by the external I/O. In this way, the first eight registers are automatically allocated to the HRK-200 attached to channel 1, while the remaining registers beginning with register 9 (up to 16 if the second rack is also an HRK-200) will be allocated to the second rack.

NOTE: *If the second rack is a CRK-300, DRK-300, or HRK-100, assign ONLY a total of 12 registers to the PDD.*

Another thing to note is that the original design intent of the PDD/PDR was to allow for real-time bus updates which, by definition, no longer occur due to the Model 650's image table operation and end-of-scan updating.

Finally, one advantage of using the Model 650 is that I/O points accessed via a PDD/PDR are now forcible.

C.3 Compatibility With Existing SY/MAX Processor Rack Addressing Configurations

Existing rack addressing configurations for Model 300, 500, and 700 systems can normally be used with a Model 650, although system characteristics may change due to the faster ladder program scan speed and image table updating technique employed by the Model 650.

From the discussion thus far, it should be apparent that existing rack addressing configurations may not represent an optimum register allocation, at least in terms of throughput. An exception to compatibility is the CRM-115/116 Bus Expander/Terminator module set, although this exception can usually be accommodated by replacing them with either PDD/PDR or LI modules, as discussed below.

Other potential exceptions are given in Section 3.6.5. Keep them in mind while reading the remainder of this appendix.

Example #1: A Model 300 in a DIGITAL I/O rack (shown below in Figure C.1).

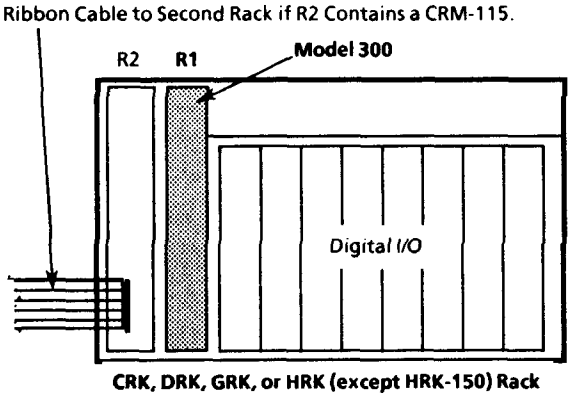


Figure C.1 Processor Location in CRK, DRK, GRK, HRK Racks

Slot 1 in Figure C.1 would typically be assigned 1,2,4,8,96,108, or 112 registers; any registers remaining (up to 112) are assigned to slot 2. If slot 2 of the CPU rack contains a module other than a CRM-115, the existing rack addressing can be used. As always, registers up to 8000 will now be available for data storage.

MODEL 300 WITH A CRM-115 IN THE CPU RACK

If slot 2 in Figure C1 contains a CRM-115, slot 1 has been assigned 112 registers, except for those rare systems which use a register module in slot R2 of the second rack.

If slot 2 of the second rack is unused, the best solution is to replace the CRM-115/116 with the PDD/PDR (Class 8030 Type EQ5138-G1 and G2). Although the rack addressing will have to be altered so that registers 1 to 4 are assigned to CPU slot 1, and registers 5 to 8 are assigned to CPU slot 2, *no* registers need be re-addressed in the ladder program. *Registers 9 to 8000 will be available as storage* registers.

If slot 2 of the second rack contains a register module, the PDD/PDR is no longer an option and an *LI/RI must be used. In the event that slot 2 of the* second rack contains a register module *other than* an LI, rack addressing will have to be adjusted to accommodate the change (the Remote Interface gets coded as drop 1, with digital I/O registers assigned to slot 1 of drop 1 while the registers for slot 2 are assigned to slot 2 of drop 1). As before, no registers need be re-addressed in the ladder program and the balance of the registers (out to 8000) are available as storage registers.

If slot 2 of the second rack contains an LI, the LI should be moved to the CPU rack to replace the CRM-115, and a Remote Interface should replace the CRM-116. Rack addressing will have to be adjusted to accommodate the change, and the existing drops will have to be re-coded. The *new* Remote Interface will become drop 1, and the other drops will have to increment by 1. The good news is that, once again, no registers will need to be re-addressed in the ladder program, as the drop re-coding will preserve their relative position.

Example #2: Model 300 in an HRK-150 rack (Figure C.2).

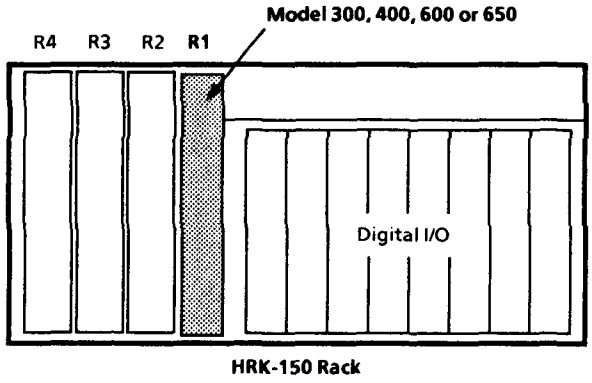


Figure C.2 Processor Location in HRK-150 Rack

By virtue of the digital I/O, slot 1 will typically be assigned 4 registers. Modules in slots 2, 3, and 4 will have registers assigned in accordance with their requirements. Existing rack addressing can be used.

Example 3: Models 300, 400, 600, 650, 500, or 700 in an RRK-type rack (Figure C.3).

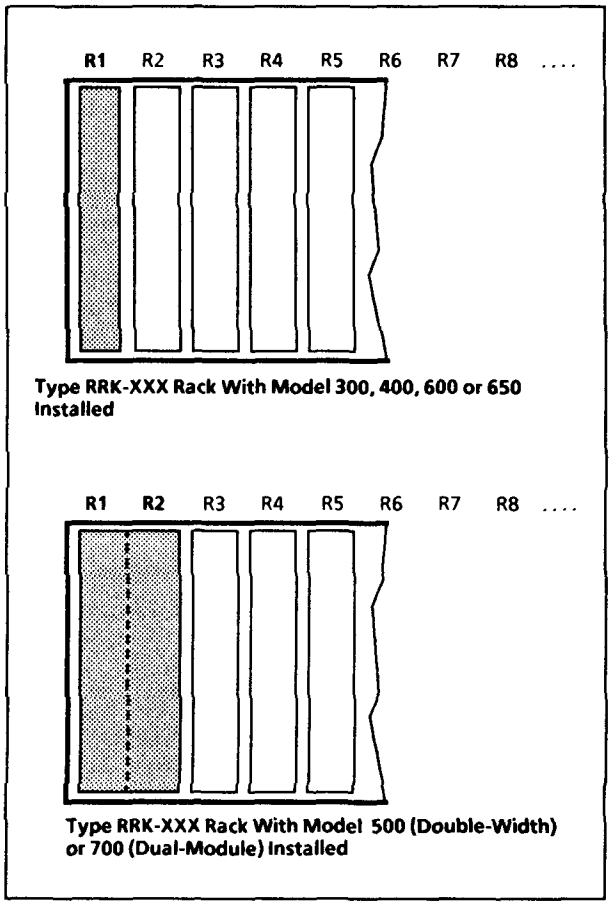


Figure C.3 Processor Location in RRK-Type Racks

The Model *500* would normally have registers assigned to slot 1; whether registers are assigned to slot 1 or not, existing rack addressing configurations will not present a problem for the Model 650, which could be used to replace any of the existing processors.

Replacing a Model 300 is the "cleanest" replacement, since like the Model 650 it is a *single-width* module. When replacing a Model 500 or 700, the question arises how to treat the newly-empty slot 2. The answer is it can either be left empty, or the rest of the modules can be moved one slot to the left, with a corresponding adjustment of rack addressing. This can be accomplished without changing register assignments in the ladder program, as the relative positions of the registers are preserved.

APPENDIX D
ETHERNET TECHNICAL INFORMATION

D.1 Overview

The unique feature of the Model 650 processor is its ability to directly connect to an IEEE 802.3a/ThinWire Ethernet Local Area Network (LAN) and communicate with other SY/MAX devices. Because the Model 650 communicates on Ethernet, a communications standard used by many vendors, it is important to understand its operating characteristics and how they affect the operation of SY/MAX devices. This Appendix is not intended as a comprehensive explanation of the theory and operation of Ethernet, but rather a review of the terminology and overall concepts as they relate to the Model 650.

D.2 Specifications

The following 802.3a specifications in Table D.1 apply to the LAN on which the Model 650 resides. The specifications for 802.3 are also included as reference for applications in which a repeater is used to connect ThinWire to ThickWire.

	802.3a	802.3
Data Rate	10Mb/s	10Mb/s
Signalling	Baseband	Baseband
Max Segment Length	185m	500m
Max Drop Cable Length	0m	50m
Max Cable segments	5	5
Max repeaters	4	4
Max Total Distance	*	*
Max nodes per segment	30	100
Min segment length	0.5m	2.5m

Table D.1 802.3a vs. 802.3 Specs

* Maximum total distance is determined by IEEE 802.3 configuration rules.

IEEE 802.3 Network Configuration Rules

CAUTION

To ensure safety and reliable data communication, the IEEE/ANSI Std. 802.3-1985 (ISO/DIS8802/2) and IEEE/ANSI Std. 802.2a-1988 standards documents, or an equivalent reference, should be consulted when laying out Ethernet networks.

Aside from the previous specifications, IEEE defines rules that must be adhered to when laying out a system. These rules deal mainly with the number of segments, the cable for these segments, and the number of repeaters to tie these segments together. These rules must be followed in order to ensure proper Model 650 communications.

Repeaters

The Model 650 will communicate through IEEE 802.3-compliant repeaters as long as the IEEE rules on the number of repeaters, cable type, and maximum segments are met.

Bridges

The Model 650 will also communicate through IEEE 802.3-compliant bridges. It will not communicate through protocol-dependent devices such as translator bridges or Gateways.

Cabling

The Model 650 has hardware on board that allows it to directly connect to ThinWire Ethernet (RG-58) cabling. The use of other cable such as ThickWire and fiber optics are only supported through the use of repeaters and bridges.

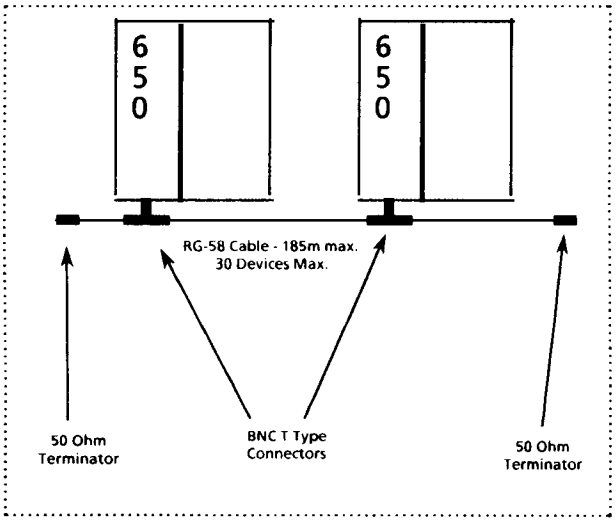


Figure D.1 Base Configuration for 802.3a ThinWire

D.3 Description

Physical Layout

An IEEE 802.3a/ThinWire Ethernet LAN consists of up to 30 devices with unique addresses connected by up to 185 meters of RG-58 cable. Each device is connected to the cable by a BNC T-type connector and the cable is terminated at each end with a 50 ohm terminator. Refer to Figure D.1. T-connectors must be at least 1/2 meter apart. Devices can be added and removed from the network without disruption of the network.

To expand a network, repeaters are added to extend beyond the 185m segment limit. The 802.3a specifications call for no more than five segments, assuming the appropriate cable types are used. Refer to Figure D.2.

To further expand, repeaters that allow for ThinWire to ThickWire and/or fiber optics Ethernet are added to a network to allow for communications between other than ThinWire devices. Using ThickWire the network can extend 500m per segment and up to 100 devices. Refer to Figure D.3. Refer to IEEE 802.3 standards for additional multimedia Ethernet connection information.

Access Method

Access to an Ethernet LAN is accomplished using a CSMA/CD (Carrier Sense Multiple Access with Collision Detection) protocol. A device on a CSMA/CD network gains access by first checking to see if any other device is currently communicating on the network. If the network is busy, the node waits until the network is available. If the network is available, the device begins to send its message to another device or group of devices. Upon completion the network is again freed for another device to send a message. If two devices begin to send a message at the same time, a collision occurs. In that case, both devices would back off for a random period of time before trying to resend. What this means to a Model 650 on an Ethernet network is that other devices and the amount of network traffic can have a significant effect on the time taken to access the network and communicate.

Other Non-SY/MAX Device Issues

Ethernet specifications define the physical and data layers for devices to communicate. That is, it provides a means by which devices can share a common signalling medium and pass messages. Messages can only be "understood", however, between devices having the same higher level protocols.

The Model 650 contains hardware and software which allow it to pass information to other devices on a ThinWire Ethernet network. But since the upper level protocol of the Model 650 is SY/MAX protocol, only those devices on the network that have the ability to interpret this SY/MAX language (currently Model 650's and VAX/VMS-based computers running Square D Class 8055 Type SFW390) are able to understand the information being transferred.

Example

Let's assume we have a ThinWire Ethernet LAN with the following devices:

5-Model 650 processors
1-DEC VAX Computer running SFW390
1-DEC VAX Computer without SFW390
2-IBM Personal computers with Ethernet interface cards installed

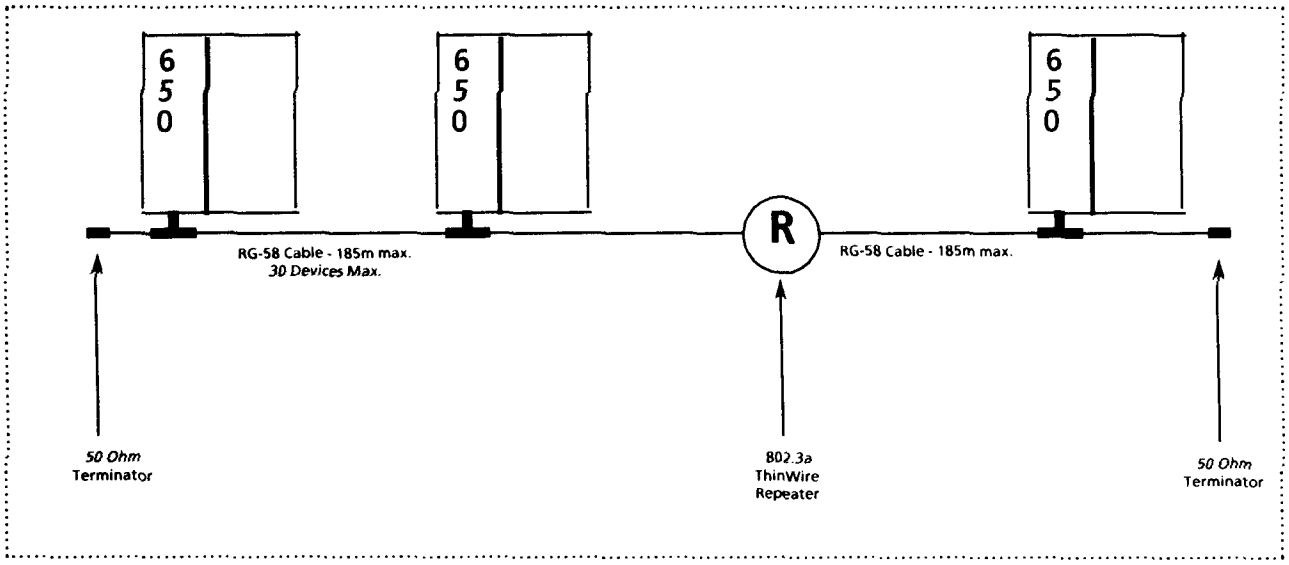


Figure D.2 Base Configuration with Repeater

The Model 650's can theoretically send messages to any device on the network. The DEC VAX with SFW390 and the other Model 650's receive and understand these messages. Even though other DEC VAX and IBM computers receive the messages and are able to tell from what node numbers they were sent, the messages are not understood and are discarded.

The reverse is also true. Non-SY/MAX devices can send messages to a Model 650, but because the Square D devices understand a different protocol, the messages are not understood and are discarded.

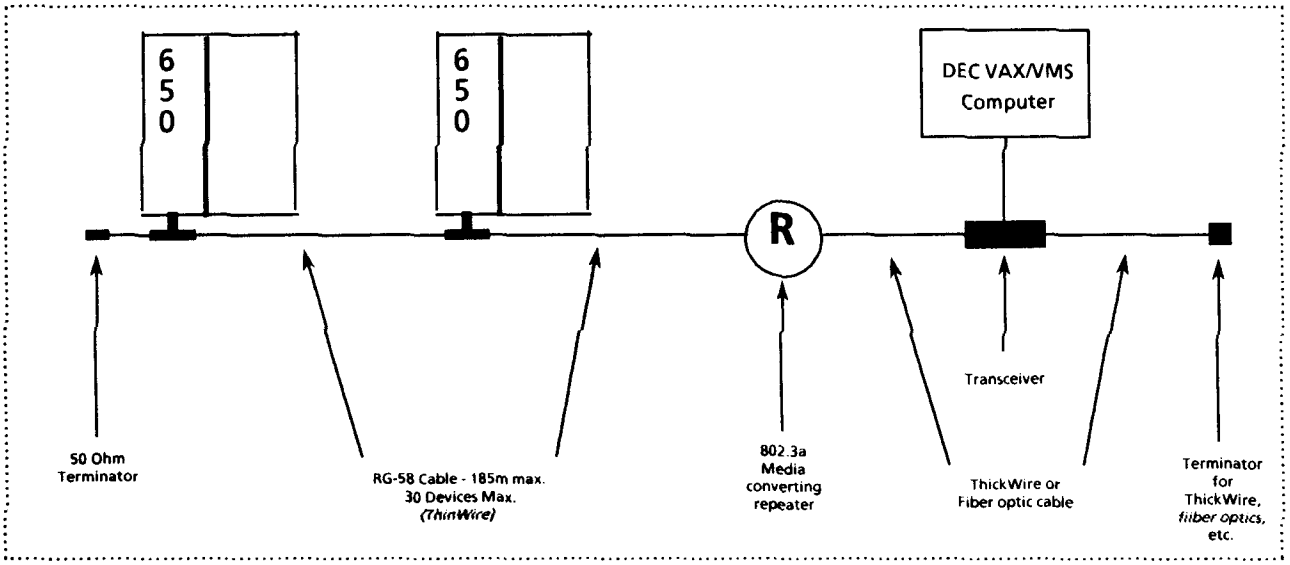


Figure D.3 Base Configuration with Repeater to ThickWire Ethernet

D.4 Throughput

The throughput of a communication system is composed of 3 basic elements.

1. The sending device's time to prepare a message and process it up to the point of its being sent out onto the network.

2. The network transfer time---the time to gain access to the network, send the message at one end, and receive it at the other end.

3. The receiving device's time to process the information and move it into the user memory.

In this case, the sending and receiving device can be either a Model 650 or a VAX computer running Class 8055 Type SFW390 software. With the Model 650, we can determine (using processor scan times) the number of registers transferred, the number of other communication packets being sent, the worst case time to prepare a message to be sent, or the time for a message to be processed after receipt.

In the case of network transfer time, 802.3a/Ethernet, like all CSMA/CD networks, is a non-deterministic means of communication. That is, the time required for a node to gain access to the network is directly dependent on the number of nodes on the network, the average length of messages being sent, etc. Because of this, the worst case time for a node to gain access to the network and send a message cannot be determined.

Because it is impossible to offer an equation to determine worst case time for a Model 650 to gain control of network, care should be taken when designing a system in which time-critical or real time control information must be transferred in a specified time period.

E CONNECTING THE VAX PROCESSOR TO THE SY/MAX MODEL 650 PROGRAMMABLE CONTROLLER

E.1 Hardware and Software Requirements

SFW390 allows SY/MAX Model 650 programmable controllers and VAX computers to effectively communicate over an Ethernet Communications Network. The SFW390 software requires that the host computer is equipped with appropriate Ethernet controller hardware and driver software.

E.1.1 HARDWARE

The SFW390 software is designed to work with any of the Digital Equipment Corporation (DEC) Ethernet controller hardware available for VAX computers. A listing of Ethernet controllers available for VAX computers, as of this writing, follows. Part numbers and installation hardware are available through DEC.

Controller	Bus Architecture
DELUA	UNIBUS
DEUNA	UNIBUS
DELQA	Q-BUS
DESQA	Q-BUS
DEQNA	Q-BUS
DEBNI	VAXBI
DEBNA	VAXBI
DESVA	MVAX2000 (ThinWire)

Other Ethernet hardware, such as cabling, transceivers, repeaters, T-connectors, terminators, etc. are also required to set up an Ethernet network. These are available from DEC and other sources.

The same cabling, T-connectors, and terminators used with a SY/MAX Model 600 programmable controller are also used to connect a Model 650 programmable controller to the ThinWire Ethernet Network. Specific information about the required cabling can be found in Section 15 and Appendix D.

E.1.2 SOFTWARE

The host computer must have appropriate Ethernet driver software. The Ethernet driver software is supplied as an integral component of VMS operating system by DEC. No special installation requirements for the Ethernet driver software currently exist. SFW390 does not require that users have a DECnet license.

E.2 Connection Diagram

Figure E.1 shows a sample network of five SY/MAX devices: three VAX computers (SY/MAX drops 3, 5 and 6) and two Model 650 programmable controllers (SY/MAX drops 10 and 15). The SFW390 software uses the Ethernet driver to communicate with programmable controllers and other VAX computers.

As shown in the diagram below, any other non-SY/MAX devices, like terminal servers and other computers, can share the Ethernet without interference. Ethernet network segments connected by a repeater are treated as a single Ethernet network for the purposes of communication between VAX computers and Model 650 programmable controllers.

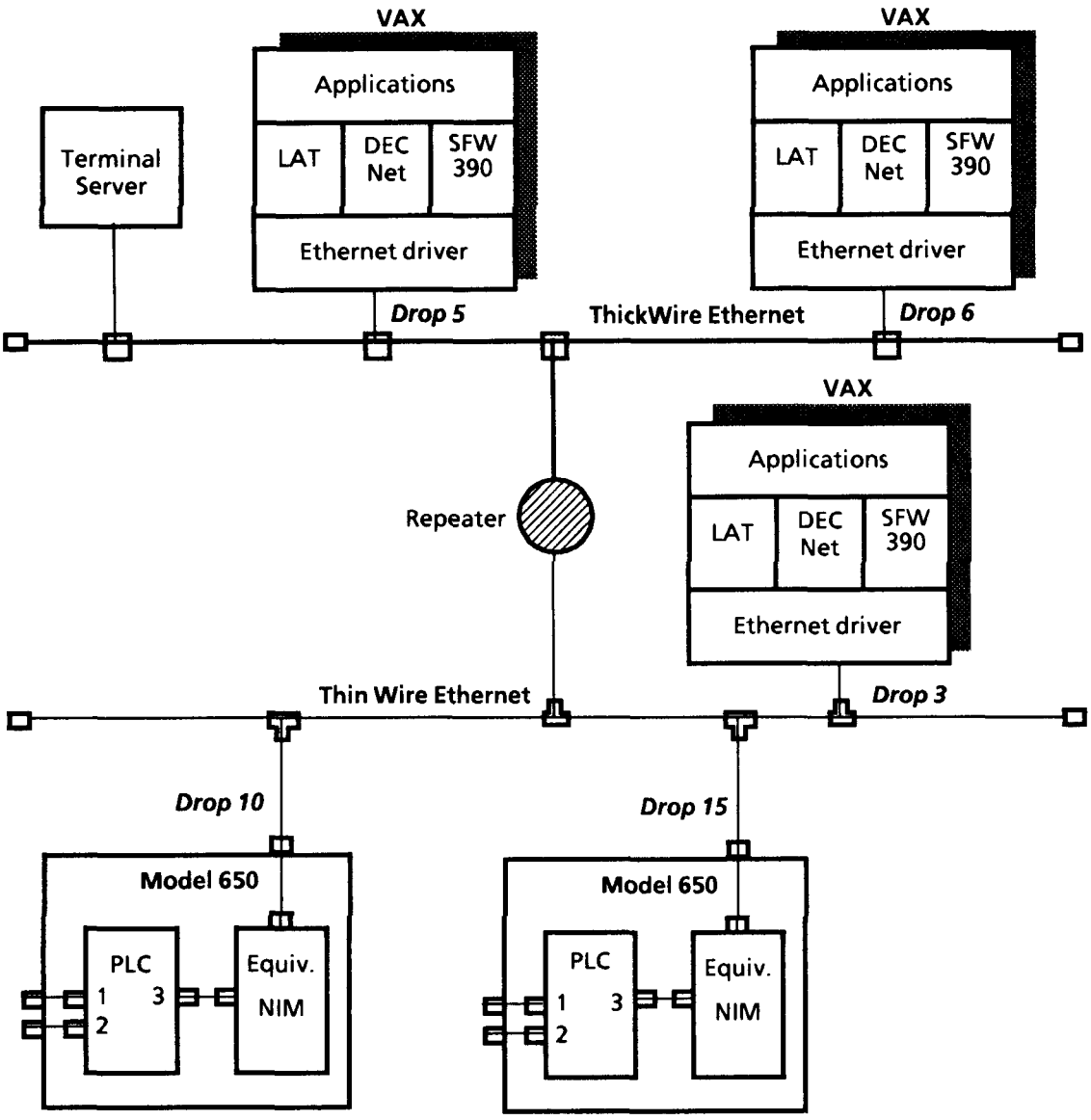


Figure E.1 - Thin Wire Ethernet Network Connected to a ThickWire Ethernet Network via a Repeater

E.3 Network Configuration Information

The Type SFW390 software supports communication between a VAX computer and up to 99 other devices, consisting of Model 650 Programmable Controllers and host computers running SFW390 (or equivalent) software, for a total of 100 devices. Devices such as terminal servers and other host computers not running SFW390 software (or equivalent) do not count towards the 100 device total. They can, however, coexist on the same Ethernet network.

Model 650 programmable controllers are connected together by means of a ThinWire (Type 10BASE2) Ethernet network. Host computers having a ThinWire (10BASE2) Ethernet interface can attach directly to this network. Host computers having a standard (10BASE5) Ethernet interface may connect to Model 650 programmable controllers through the use of an appropriate repeater. Refer to Figure E.1.

The IEEE/ANSI Std. 802.3-1985 (ISO/DIS 8802/2) and IEEE/ANSI Std. 802.3a-1988 specify a maximum of 100 stations per 10BASE5 coax segment and a maximum of 30 stations per 10BASE2 coax segment. Through the use of repeaters, it is possible to have a large number of stations on an Ethernet network. Minimum cable length between two stations on a 10BASE2 coax segment is 2 feet (approximately 0.5 meter). The maximum distance between any two stations on a 10BASE2 coax segment is limited to 607 feet (approximately 185 meters), in accordance with the above specifications.

More information about the Class 8055 Type SCP65X Programmable Controller as well as cabling information and specifications can be found in Section 15 and Appendix D.

CAUTION

To ensure safety and reliable data communication, the IEEE/ANSI Std. 802.3-1985 (ISO/DIS8802/2) and IEEE/ANSI Std. 802.2a-1988 standards documents or an equivalent reference should be consulted when laying out Ethernet networks.

E.4 Routing Information

Standard SY/MAX routing is used with the SFW390 software and the Model 650. Routing is the data instruction that controls how messages are communicated from one device to another. A route actually consists of a route block structure, but is identified by a user as a series of numbers separated by commas. The series of commas which comprise a route include the numbers that identify the sender of a data message (this can be a specific task running in a VAX computer, or it can be a specific Model 650 programmable controller) and the numeric identifier of the intended destination location (this can be a specific task in a VAX computer, a certain Model 650 programmable controller on the SY/NET network, a global queue, mailbox register, or Ethernet configuration register).

Route values of (000-199) are the general route values used to direct a communication to specific devices on the network. However, above this range, are three special route values that may also be used; route 200, route 201 and oute 204. Route 200 directs the communication to the equivalent NIM within the VAX computer or the programmable controller. Route 201 indicates a "whoever-I-am" source route and route 204 indicates a "don't care" route and is typically ignored as if it were not there. These special routes are explained in greater detail in Section E.4.3 "Special Routing Considerations" and Section 15.4.2.

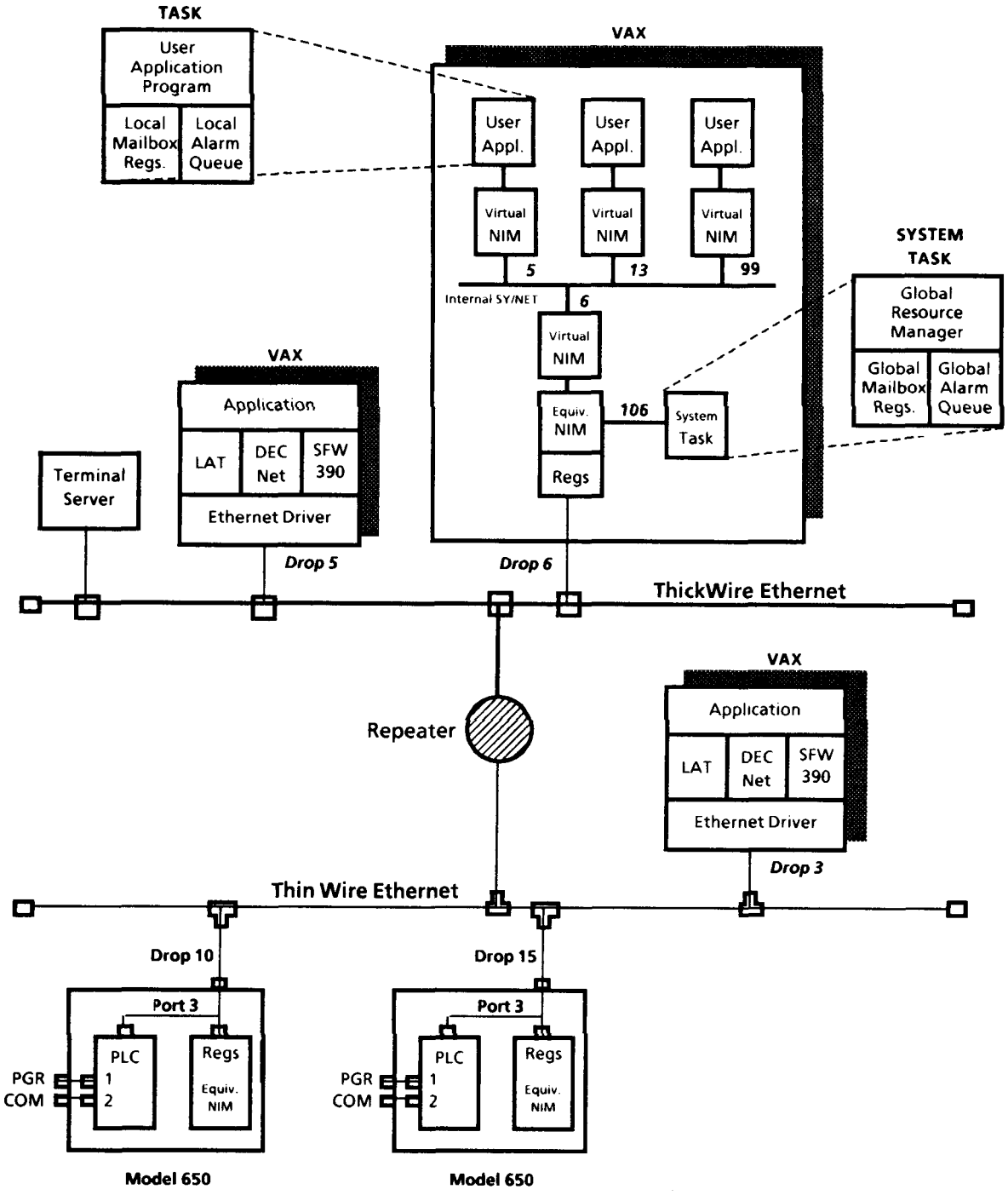


Figure E.2 - Possible Ethernet Network Configuration with Model 650 Programmable controllers and VAX Computers Running SFW390; Including Illustrations of VAX Architecture

E.4.1 VAX-INITIATED MESSAGING

Depending upon your operation, the routing from source device to destination device is done differently. The following tables describe the methods used to route communications for each of three categories of VAX computer initiated message transmissions. These categories include local register Read and Write operations, Reads and Writes to global mailboxes or global alarm queues, and Reads and Writes to Ethernet Configuration registers. Notes referenced within each of the tables can be found in Section E.4.3. Also, Figure E.2 on the previous page can be used as a reference for the examples provided with each explanation. In Figure E.2 the enlargement representing VAX 6 illustrates the internal architecture of the tasks, mailboxes, and alarm queues.

When performing local register Read and Write operations:

	FIRST ROUTE #	SECOND ROUTE #	THIRD ROUTE #	FOURTH ROUTE #
1. From a VAX Computer to the same VAX Computer:	Source task number	Destination task number	None	None
Examples: Task 5 on VAX 6 reads registers from Task 99 on VAX 6 Task 1 on VAX 3 reads from its own local mailbox registers	5, See note #5 on page E-12.	99		
2. From a VAX Computer to a second VAX Computer:	Source task number	Source VAX number	Destination VAX number	Destination task number
Example: Task 1 on VAX 3 reads registers from Task 13 on VAX 6	1,	3,	6,	13
3. From a VAX Computer to a Programmable Controller:	Source task number	Source VAX number	Destination programmable controller	None
Example: Task 5 on VAX 6 writes to programmable controller registers in Model 650 #15	5,	6,	15	

When performing Read and Write operations with global mailboxes and alarm queues:

	FIRST ROUTE #	SECOND ROUTE #	THIRD ROUTE #	FOURTH ROUTE #
1. From a VAX Computer to the same VAX Computer:	Source task number	Source VAX number	Source VAX number plus '100' (see note 1)	None
Example: An unspecified task on VAX 3 reads from its own global mailbox registers	201, (see note 4)	3,	103	
2. From a VAX Computer to a second VAX Computer:	Source task number	Source VAX number	Destination VAX number plus '100' (see note 1)	None
Example: Task 4 on VAX 3 writes an alarm to a global alarm queue on VAX 6	4,	3,	106	
3. From a VAX Computer to a Programmable Controller:	Source task number	Source VAX number	'200' (see note 2)	Destination programmable controller
Example: Task 4 on VAX 5 writes to the Equivalent NIM mailbox registers of Model 650 #15	4,	5,	200,	15

When performing Reads or Writes to Ethernet configuration registers

	FIRST ROUTE #	SECOND ROUTE #	THIRD ROUTE #	FOURTH ROUTE #
1. From a VAX Computer to the same VAX Computer:	Source task number	'200' (see note 3)	VAX number	None
Example: Task 1 on VAX 3 reads from its own Ethernet configuration registers	1,	200,	3	
2. From a VAX Computer to a second VAX Computer:	Source task number	Source VAX number	'200' (see note 2)	Destination VAX number
Example: Task 1 on VAX 3 reads writes to the Ethernet configuration registers of VAX 5	1,	3,	200,	5
3. From a VAX Computer to a Programmable Controller:	Source task number	Source VAX number	'200'	Destination programmable controller
Example: Task 12 on VAX 5 reads from the Ethernet configuration registers of Model 650 #10	12,	5,	200,	10

E.4.2 MODEL 650 PROGRAMMABLE CONTROLLER INITIATED MESSAGING

Depending upon your operation, the routing from source device to destination device is done differently. The following tables describe the methods used to route communications for each of three categories of programmable controller initiated message transmissions. These categories include local register Read and Write operations, Reads and Writes to global mailboxes or global alarm queues, and Reads and Writes to Ethernet Configuration registers. Notes referenced within each of the tables can be found in Section E.4.3. Also, Figure E.2 on page E-4 can be used as a reference for the examples provided with each explanation. In Figure E.2 the enlargement representing VAX 6 illustrates the internal architecture of the tasks, mailboxes, and alarm queues.

When performing local register Read and Write operations:

	FIRST ROUTE #	SECOND ROUTE #	THIRD ROUTE #	FOURTH ROUTE #
1. From a Model 650 programmable controller to a VAX Computer:	Source programmable controller	Destination VAX number	Destination task number	None
Example: Model 650 #15 reads mailbox registers from Task 7.	15,	5,	7	
2. From one Model 650 programmable controller to another:	Source programmable controller	Destination programmable controller	None	None
Example: Model 650 #15 writes to registers in programmable controller #10.	15,	10		

When performing Read and Write operations with global mailboxes and alarm queues:

	FIRST ROUTE #	SECOND ROUTE #	THIRD ROUTE #	FOURTH ROUTE #
1. From a Model 650 programmable controller to a VAX Computer:	Source programmable controller	Destination VAX number plus '100' *(see note 1)*	None	None
Example: Model 650 #10 writes an alarm to a global alarm queue on VAX 5.	10,	105		
2. From one Model 650 programmable controller to another:	Source programmable controller	'200'	Destination programmable controller	None
Example: Model 650 #15 writes to Equivalent NIM mailbox registers in programmable controller #10.	15,	200,	10	

When performing Read and Write operations to Ethernet configuration registers

	FIRST ROUTE #	SECOND ROUTE #	THIRD ROUTE #	FOURTH ROUTE #
1. From a Model 650 programmable controller to a VAX Computer:	Source programmable controller	'200' (see note 2)	Destination VAX number	None
Example: Model 650 #15 reads Ethernet configuration registers from VAX 6.	15,	200,	6	
2. From one Model 650 programmable controller to another:	Source programmable controller	'200' (see note 2)	Destination programmable controller	None
Example: An unspecified programmable controller reads Ethernet configuration registers from the equivalent NIM of Model 650 #15.	201, (see note 4)	200,	15	

E.4.3 SPECIAL ROUTING CONSIDERATIONS

Note 1 -

When the destination of a route instruction is a VAX computer's global queue or mailbox, the destination drop number is 100 plus the VAX computer's drop number. For example: The global alarm queue in VAX 6 would be specified as 106, and if task 10 on VAX 3 writes to the global alarm queue in VAX 6, the entire route would be: 10, 3, 106.

Note 2 -

Route 200 is a special route whose function indicates that registers in the Equivalent NIM of a programmable controller, or the Ethernet configuration registers in the VAX are to be read in the form of register data. For example: When reading from the Ethernet configuration registers in VAX 6, the destination route instruction will be ..., 200, 6; and when reading from the equivalent NIM in programmable controller 10, the destination route instruction will be ..., 200, 10.

Note 3 -

When a Task within a VAX reads from the Ethernet Configuration registers within that same VAX, the route consists of the source task number, followed by the number 200, followed by the VAX number. For example: When Task 4 on VAX 5 reads from the Ethernet Configuration registers within VAX 5, the route is: 4, 200, 5.

Note 4 -

The route instruction '201' can be used in the source route when the initiating task chooses to communicate over the network without specifying its own route address. For example: If an unspecified task on VAX 3 reads from programmable controller #15, the route is: 201, 3, 15.

The special 201 route, which can only be used in the first routing instruction (with one exception) can replace either a task number (VAX initiated message) or a programmable controller number (programmable controller initiated message). The 201 route can appear in the last routing instruction only when the destination drop is the same as the source drop (for example, 201, 200, 201 from a programmable controller will read its Equivalent NIM registers).

Route 201 is sometimes referred to as the "whoever I am" route.

Note 5 -

When a task reads its own local mailbox, it reads an empty route, which is signified by "255" in the first route number.