4.0 Control Tables and Files

4.1 Introduction

Inside the --------, battery-backed static RAM stores and retains the downloaded tables or files which determine the operation, and personality of the --------. The -------- uses configuration tables to monitor, activate, or adjust its corresponding components and features. Each definable, separate table is then grouped or assembled into a batch. This batch of tables can be transferred to a computer to be viewed and or modified with ------- 64. The tabular information distinctly controls -------- character and performance.

With the exception of the XXXCODE program binary and the macro key binary files, all the files are ASCII type. The following are the tables/files used and their extensions.

Note: Before changing any information within the original files, use good operating procedures and make a backup.

Table 4-1 – XXXCODE Tables and File Types

Table	File Extension	File Type	Description
XXXCODE program	.src	ASCII	Contains the application specific control program
	.sr5	binary	XXXCODE assembler output, created during download; also disassembled during upload by ---------- 64
Configuration	.cfg	ASCII	Determines -------- general settings and parameters
Network Connection	.net		Contains data for -------- to communicate with other units
Polling	.pol		List of registers that should be polled from other units
Setpoint	.stp		List of setpoint values
Filter Constant	.flt		List of analog input filter constants
Calibration	.cal		Controls conversion of analog I/O values to engineering units

Table	File Extension	File Type	Description
Macro	.mac	ASCII	Specifies action to be executed when macro keys are pressed
	.ma5	binary	Output of the macro assembler; must be created for download. Also created by uploading a macro table from the --------
Tag	.tag	ASCII	Tag names, units, descriptions associated with registers
LED	.led		Mapping of digital inputs/outputs, alarms, and bar graphs to LEDs
Precalibration	.pre		Adjusts analog inputs/outputs for accurate readings
Archive Array	.arc		Sets up -------- to automatically store data in archive arrays
Quiescent Telemetry	.qtl		List of registers that should be sent on time or should change
Modbus®	.mod		If a Modbus® slave device, provides floating point register values and register index mapping

All of these tabular files can be uploaded from, or downloaded to, the -------- using ---------- 64 (see *Chapter* ***Error! Reference source not found.*** *– -------- 64 IDE*). In addition, you can change a program, macro program, configuration table, setpoints, filter constants, calibration, and pre-calibration from the keypad.

ASCII text comprises table files and may have imbedded comments and whitespace to facilitate readability. Comments consist of text following a colon character (:). They can be on an individual line or at the end of a data line entry. Whitespace can consist of blank lines, spaces, and tab characters. Whitespace can be used freely, except inside literal strings (enclosed in quotes), where it is interpreted literally.

All -------- tables and programs are backward-compatible with the entire *<company>* PLC family, including the ----------, --------, and --------.

4.2 Configuration Table

The configuration table contains several -------- settings and options and is stored in write-protected RAM. It can be viewed or changed from the keypad, or by using the ---------- 64 program.

4.2.1 View the Configuration Table On the --------

You can use the SPECIAL > EXAM CFG menu option to view the configuration table on the ---------- display.

1. To view a particular configuration table field, first use the ▲▼ keys to select the desired category. (Category titles are shown as major headings in the sections that follow.)
2. Press [ENTER] once you have selected the desired category. Select the desired field and then press [ENTER] again. The current field value is displayed.

Press [EXIT] (or any key) to stop viewing the selected field and return to the category field list.

To move to another category, press [EXIT] again, and then use the ▲▼ keys.

Press [EXIT] again to exit the configuration table viewing function.

4.2.2 Change the Configuration Table From the Keypad

The -------- must be in standby mode in order to change the configuration table. To change the table from the keypad, use the CHNG CFG option, or use the <*keyword*>.8 shortcut (e.g., 10.8).

To change a field from the keypad:

1. Press the ▲▼ keys to select the category, and then press [ENTER].
2. Select the desired field from the category field list, and then press [ENTER] to view/edit the field.
3. Press the ▲▼ keys to select a different value. For numeric fields, key in a number directly from the 32-key keypad.
4. Press [ENTER] to accept the value, or press [EXIT] to retain the original value.
5. Select another field to change, or press [EXIT] again to select a different category.

Once you have made all desired changes to the configuration table:

6. Press [EXIT] until the SAVE–YES prompt is displayed.
7. Press the ▲▼ keys to select either SAVE–YES or SAVE–NO.
8. Press [ENTER] when the desired choice is displayed, or press [EXIT] to return to editing the configuration table.

If you select SAVE–YES and then press [ENTER] in steps 7 – 8, the -------- scans the new configuration table for any errors or conflicts. If it finds one, the offending field is displayed to facilitate changing its value and resolving the problem.

Note: Sometimes more than one field is involved, for a few rules exist to enforce a certain relationship between two fields (defined in the field descriptions within this chapter).

4.2.3 Configuration Table Fields

You can set many configuration table fields. Major topic headings correspond with category names in the table, whereas minor headings correspond to the field names.

-------- Information (PLC.INFO)

-------- *Name (PLC.NAME)*—Use this 35-character text field to easily identify the --------.

To change the -------- name from the keypad:

1. Press the ▲▼ keys to select each desired letter in sequence. Press [ENTER] after each desired letter.
2. Press [EXIT] when done**.** Doing so saves your changes in this instance.

This text is displayed as a banner by accessing the SPECIAL > PLC.INFO menu option or by pressing [STATUS] [ENTER].

Digital Outputs/Inputs (DO/DI)

Total Number of Physical DOs (PHYS.DO) – Set to correspond with the number of physical digital outputs installed on the --------. Value range: 0 – 192.

Number of Physical DIs (PHYS.DI) – Set to correspond with the number of physical digital inputs installed on the --------. Value range: 0 – 390.

Number of Internal DIs (INTRN.DI) – Determines the number of internal digital inputs (programming flags) available to the XXXCODE programmer. These DIs do not have corresponding physical hardware. Value range: 0 – 10000.

Internal DI Starting Index (INT.DI.IDX) – Determines the starting index of internal DIs to be reported via -------- protocol telemetry and displayed on the --------'s LED annunciator panel. Value range: 0 – 10000.

Note: Leave a large numerical gap between the last physical and first internal DIs, such that first internal DI starts at an easily-identifiable round number (e.g., 500). In this way, the -------- does not waste time reporting *phantom* DIs in the unused range between the last physical DI and the first internal DI. Phantom DIs are not reported via -------- protocol telemetry, and cannot be mapped to the -------- LED display.

Analog Inputs/Outputs/Setpoints (AI/AO/SP)

Number of Physical AIs (PHYS.AI) – Set to correspond with the number of physical analog inputs installed on the --------. Value range: 0 – 112.

Number of Internal AIs (INTRN.AI) – Determines the number of internal analog inputs available to the XXXCODE programmer. These registers hold intermediate calculation values and the like. They have no corresponding physical input. Value range: 0 – 10000.

Number of Physical AOs (PHYS.AO) – Set to correspond with the number of physical analog outputs installed on the --------. Value range: 0 – 60.

Number of Internal AOs (INTRN.AO) – Determines the number of internal analog outputs available to the XXXCODE programmer. These registers hold intermediate calculation values and the like. They have no corresponding physical input. Value range: 0 – 10000.

Number of Setpoints (SP) – Determines the number of setpoints available to the XXXCODE programmer. These registers hold constant values (e.g., pump turn-on and turn-off points). Value range: 0 –3000.

Number of Index Registers (INDEX) – Determines the number of index registers available to the XXXCODE programmer; used to index into arrays or for internal calculations. Value range: 0 – 10000.

Input Scanning (INPT.SCAN)

You can select several different input scanning methods using the following parameters. Inputs can be scanned in synchrony with the XXXCODE program, or based entirely on time.

Scan at the Top of XXXCODE Program (SCAN.TOP) – This yes/no value determines whether or not inputs are scanned each time the top of the XXXCODE control program is encountered. Default value: NO.

XXXCODE Line to Trigger Scanning (SCAN.LINE) – Defines the XXXCODE line number which triggers input scanning when executed. It only applies if the SCAN.TOP value is YES. Value range: 0 – 65534.

Input Scan Rate (SCAN.RATE) – Defines the rate (in *ms*) at which input scanning occurs. It only applies if the SCAN.TOP value is NO. Each time a XXXCODE line is executed, the -------- checks SCAN.RATE. If its number of milliseconds has elapsed since the last scan, an input scan is performed. Value range: 0 – 65535, where 0 = as fast as possible (once for each executed XXXCODE line.

Input Scan AI Count (AI.COUNT) – Specifies the number of analog inputs to scan each scan cycle. (All digital inputs are scanned each cycle.) Value range: 0 – <*number of physical AIs*>.

Timers/Counters (TMR/CTR)

Number of Pulse Counters (PULSE.CTR) – Determines the number of pulse counters, each being associated with a physical digital input (up to the first 16 DIs). Pulse counters count the number of pulses sensed on the corresponding DI. Value range: 0 – 16.

Pulse Counter Trigger Map (PULSE.MAP) – Determines the trigger edge of the pulse counters. This is interpreted as a 16-bit binary value, where each bit (0 or 1) determines the trigger edge of the associated pulse counter. Bit 0 corresponds with the first pulse counter, bit 1 corresponds with the second pulse counter, etc.

Value range: 0 – 65535, where 0 (all bits 0) means all pulse counters trigger (count) on a 1-to-0 transition. A value of 65535 (all bits 1) means all pulse counters trigger on a 1-to-0 transition.

Pulse Filter Delay (PLS.DLY.0, ... 15) – There are 16 pulse filter delay fields, corresponding to up to 16 pulse counters. A pulse filter delay, expressed in milliseconds, determines the minimum pulse width sensed by a pulse counter. A low value causes a pulse counter to respond to high frequency pulses. A high value is useful to filter out high frequency noise (for instance contact bounce) on a pulse source.
Value range: 0 – 65535, where 0 means no filtering. Default value: 100.

Number of Hours Timers (HOUR.TMR) – Determines the number of hour timers.
Value range: 0 – 256. Default value: 40.

Number of HMS Timers (HMS.TMR) – Determines the number of HMS (hours-minutes-seconds) timers. Value range: 0 – 256. Default: 10.

Number of Event Counters (EVENT.CTR) – Determines the number of event counters.
Value range: 0 – 256. Default: 20.

Number of Seconds Timers (SEC.TMR) –*Determines* the number of seconds timers.
Value range: 0 – 256. Default: 40.

Special Function Registers (SPEC.REGS)

Use the fields in this category to determine which -------- registers are used to perform various special functions. The default is adequate in most cases, but being able to remap registers may prove useful if the default is needed for something else.

AO for Day of Week (WK.DAY.AO) – This defined AO field stores the day of the week (1 – 7, where 1 is Sunday), when the RTC (real-time clock) option is installed. Default: AO 93.

AO for Day of Month (MNTH.DY.AO) – This defined AO field stores the day of the month (1 – 31), when the RTC (real-time clock) option is installed. Default: AO 94.

AO for Month of Year (MONTH.AO) – This defined AO field stores the month of year (1 – 12), when the RTC (real-time clock) option is installed. Default: AO 95.

AO for Year (YEAR.AO) – This defined AO field stores the year (00 – 99), when the RTC (real-time clock) option is installed. Default: AO 96.

Time of Day (TIME.AO) – This defined AO field stores the current time, in 24-hour *hours timer* format (hours, fractions of hours), when the RTC (real-time clock) option is installed. Default: AO 90.

Debug Statement Number Setpoint (DEBUG.SP) – The setpoint defined by this field stores a XXXCODE program line number such that, when the line executes, the SBY LED flashes. This is useful for developing and testing XXXCODE programs. Default: SP 125.

Telemetry Control Setpoint (TLM.CTL.SP)

Use the setpoint defined by this field to control telemetry operation and monitoring. Default: SP 126.

Table 4-2 lists the different setpoint values and the effect each has:

Table 4-2 – Telemetry Control Setpoint Values

Value	Effect
0	Normal 0peration
1	Suppress communication fail (CF) messages on alphanumeric display
2	Monitor network activity to alphanumeric display in FROM.TO.TYPE format
4	Suppress quiescent telemetry message origination
8	Suppress polling telemetry message origination

These setpoint values can be added together to have multiple effects; e.g., a value of 6.0 is read as *monitor to the alphanumeric display* and *suppress quiescent telemetry*. A value of 12.0 is read as *suppress polling telemetry* and *suppress quiescent telemetry*.

Communications Fail DI (CM.FAIL.DI) – This defined digital input register field is used to indicate that a communications failure has occurred between this unit and another (in which event the DI is turned ON). Default: DI 863.

Communications Fail Address AO (CM.FAIL.AO) – This defined analog output register field is used to indicate the network address of the unit that last declared a communications failure (0, if this unit). Default: AO 91.

Address of Unreachable Unit AO (UNRCH.AO) – This defined analog output register field is used to indicate the network address of the unit that cannot be reached during a communications failure. Default: AO 92.

Power Count AO (PWR.CNT.AO) – This defined analog output register field is used to indicate the cumulative number of power interruptions to the -------- unit. The PWR.CNT.AO

and the RST.CNT.AO values should reflect the same count for proper operation. Check the log file if these counts are ever different. Default: AO 97.

Reset Count AO (RST.CNT.AO) – This defined analog output register field is used to indicate the cumulative number of system resets to the -------- unit. The PWR.CNT.AO and the RST.CNT.AO should values reflect the same count for proper operation. Check the log file if these counts are ever different. Default: AO 98.

Random Number AO (RND.NUM.AO) – This defined analog output register field automatically generates a random number (0.0 – 99.9) each time it is read. Default: AO 99.

Network ID and Node Address AO (NET.ADR.AO) – This defined analog output register field holds the -------- network ID and node address. It can be read by XXXCODE or examined from the keypad. The format of the data stored in this register is:

```
AO 100 = NetID * 1000 + NodeAddr
```

For example, a value of 123045 means network ID 123 and node address 45.

Default: AO 81.

Run mode DI (RUN.DI) – This defined digital input field is on while the -------- is in RUN mode and off when in STANDBY. Default: DI 876.

Program Line To Go To On Interrupt (SW.INT.LN) – When an interrupt is received on the first interrupt DI, the -------- jumps to the XXXCODE program *<line>* defined in this field. Each successive interrupt DI jumps to *<line>* + 100. For example, if an interrupt is received on the fourth interrupt DI, program execution jumps to the SW.INT.LN value + 300. Default: 10000.

Interrupt Enable DI (SW.INT.DI) –Interrupt DIs are enabled if this defined digital input field status is ON; otherwise they are disabled. Default: DI 878.

DI To Flag Keypad ALARM ACK (ACK.KEY.DI) – This field defines the digital input (normally internal) which indicates that the user has acknowledged alarms through the keypad. When the user presses the [ALARM ACK] key or selects the ALRM ACK menu function, this DI is set to the ON state. Default: DI 862.

Alarm Acknowledge DI Enable (DI.ACK) – This ON/OFF setting enables/disables the Alarm Acknowledge DI. Default: ON.

Alarm Acknowledge DI (ALM.ACK.DI) – This field defines the digital input (normally physical) used for an alarm acknowledge function. Typically this DI is wired to a pushbutton. Default: DI 0.

Security Threat DI (SECURE.DI) –*Determines* which DI (normally internal) to use to flag a potential security threat—defined as three successive attempts within one minute to unlock the -------- using an incorrect keyword. If this occurs, the selected DI is turned ON and all keywords are locked out for five minutes. Default: DI 871.

AC Fail DI (AC.FAIL.DI) – Default: DI 872.

Trace Delay Index Register (TRACE.IR) – This register holds the trace delay index which determines XXXCODE execution speed. Slowing execution lets program code be reasonably monitored using one of the trace functions.

The register holds the number of 1 ms time units allowed per line of XXXCODE. For example, if 50 is stored and a XXXCODE line takes 20 ms to execute, then the -------- delays for 30 ms (that is, 50 – 20) before advancing to the next line. Default: IR 73.

Trace Level 1 DI (TRACE1.DI) – Stores the index of the DI register used to enable/disable tracing of XXXCODE execution line number information (level 1 information). When this DI is ON, the following data is sent to the standard output device:

- *nnnn* – the XXXCODE line number executed
- A – the ACTION part of the expression has been executed
- E -the ELSE part of the expression has been executed

In addition, if a XXXCODE line number is in the debug setpoint (DEBUG.SP, default SP 125) and the TRACE1.DI is ON, additional data about the line number is reported. Default: DI 874.

Trace Level 2 DI (TRACE2.DI) – Stores the index of the DI register used to enable/disable additional XXXCODE program line execution tracing (level 2 information). Default: DI 875.

Card Count AO (CRD.CNT.AO) – Determines the analog output that stores the total number of installed -------- I/O cards—not including the processor card. Default: AO 80.

Card Health DI (HEALTH.DI) – Determines the first digital input assigned to indicate the operational status of an I/O card. The next consecutive *X* digital inputs are used to indicate the status of other cards, where *X* is the number of installed I/O cards. An ON state indicates a card is functioning, while an OFF state indicates a fault. Default: DI 887.

Transmit Queue Overflow (XMT.QUE.AO) – This register counts every transmit queue overflow. SCADA can monitor this value to learn if the L2000/-------- ever clears any transmit queue because too many entries have been added to it. The Port N.Q.SIZE can be increased and/or communications can be decreased on the port so that XMT.QUE.AO is not incrementing. Default: AO 82.

Network/Telemetry Items (NET/TLM)

Network ID (NET.ID) – Determines the -------- network ID when using Data Express Plus messaging. Value range: 1 – 255. Default: 1.

Network Node Address (NODE.ADR) – The unique network node address, used to identify this -------- so that -------- (--------), ---------- (-------- & --------), and Modbus® protocol telemetry messages can be directed to it. Value range: 1 – 255. Default: 1.

Quiescent Operation Startup Delay (Q.STRT.DLY) – On power up, determines how long the -------- waits before it begins sending values from its quiescent table. It also applies to changes from standby to run modes, and when quiescent operation is re-enabled using the telemetry

control setpoint. This value uses 10 ms time units. Value range: 0 – 65535. Default: 500 (5 seconds).

Quiescent Telemetry Table Scan Rate (Q.SCN.RATE) – Determines how often the quiescent table is scanned to check if any changed values need to be sent. Interval-based entries are serviced at the appropriate time, regardless of this field. This value uses 10 ms time units. Value range: 0 – 65535, where 0 means scan as often as possible. Default: 100 (1 seconds).

Percentage of Quiescent Threshold (PCNT.THRS) – This field applies only to threshold-based quiescent communications with a peer system on a dialup link. If a quiescent message is sent to such a system while a connection exists, the -------- tries to take advantage of the connection depending on this field value. For example, if set to 25%, any quiescent threshold-based points changed by more than 25% are sent at that time as well. A setting of 100% effectively disables this feature, as it requires a full threshold change before sending the quiescent point. Value range: 10 – 100%. Default: 100%.

Percentage of Quiescent Interval (PCNT.INTV) – This field applies only to interval-based quiescent communications with a peer system on a dialup link. If a quiescent message is sent to such a system while a connection exists, the -------- tries to take advantage of the connection depending on this field value. For example, if set to 60%, any quiescent points with a normal interval elapsed by more than 60% of are sent at that time. A setting of 100% effectively disables this feature, as it requires a full elapsed interval before sending the quiescent point. Value range: 10 – 100%. Default: 100%.

Power Up Quiescent Table Dump (PWR.UP.DMP) – Determines if the entire quiescent table is dumped (transmitted) when the -------- is powered on. It also applies to changes from standby to run mode, and when quiescent operation is re-enabled using the telemetry control setpoint. If set to NO, only quiescent entries that cross their threshold from that point forward, or whose interval expires, are sent out. Value range: Yes/No. Default: YES.

Write Through Polling (POLL.WRIT) – Enables/disables write-through polling. Value range: Yes/No. Default: YES.

Event Logging (LOGGING) – Determine what level of event logging messages are sent to the various output devices. Event log messages are generated by various system events, with each message being assigned a level of importance.

Table 4-3 – Event Logging Levels

Term	Definition
NONE	No logging
VERBOS	Verbose messages
INFO	Information messages
WARN	Warning messages
ERROR	Error messages

An INFO setting logs informational, warning, and error messages for a given device. A NONE setting disables logging to that device.

Alphanumeric Display Logging Level (ALPHA.LVL) – Determines the level of message logging for the alphanumeric display. Default: ERROR.

RAM Logging Level (RAM.LVL) – Determines the level of message logging to the memory (RAM) log. Default: INFO.

Number of Log Messages to Keep in RAM (NUM.MSG) – Determines the number of messages kept in the RAM message log. Value range: 128 – 20000. Default: 1000.

Allow RAM Log Messages to Wrap (RAM.WRAP) – Determines what happens when the RAM log is full and a new message is generated. If set ON, the oldest message is discarded to make room for the new message (FIFO). If set OFF, no new messages are stored until the log is cleared (using SPECIAL > DIAGNOST > CLR LOG function).

Activity Monitoring (MONITOR)

These settings determine which activity monitoring messages should be generated. More than one type of monitoring can be enabled at a time.

For all settings in this category (unless specified otherwise):

Generated output is sent to the standard output device (defined by the STD.OUT setting).

Value range: YES/NO. Default: NO.

These settings are for <*company*> use only and should be used with care, as they can affect system operation.

Monitor Analog Input Scanning (SCAN) – Determines if analog input scanning activity is monitored.

Monitor Self Test (SELF.TEST) – Determines if self-test activity is monitored.

Monitor Network Traffic Addressed to Us (NET.US) – Determines if network/telemetry messaging addressed to this unit is monitored. Possible settings are NONE, or any combination of P1, P2, P3, or P4 to specify the port(s) for which monitoring is to occur.

Monitor Network Traffic Addressed to Any Unit (NET.ALL) – Determines if network/telemetry messaging addressed to any unit is monitored. Possible settings are NONE, or any combination of P1, P2, P3, or P4 to specify the port(s) for which monitoring is to occur.

Monitor Network Retries (NET.RETRY) – Determines if network/telemetry retries (ON, BUSY, or NO ACK conditions) for this unit are monitored. Possible settings are NONE, or any combination of P1, P2, P3, or P4 to specify the port(s) for which monitoring is to occur.

Monitor Network Traffic, Raw Mode (NET.RAW) – Determines if network/telemetry messaging (for all units heard) is monitored in raw mode, showing the raw data bytes transmitted and received. Possible settings are NONE, or any combination of P1, P2, P3, or P4 to specify the port(s) for which monitoring is to occur.

Monitor Network Transmit Queue (NET.QUEUE) – Determines if transmission queue activity is monitored, showing each time a message is added to or removed from the transmission queue. Possible settings are NONE, or any combination of P1, P2, P3, or P4 to specify the port(s) for which monitoring is to occur.

Monitor Dialup Modem Activity (DIAL) – Determines whether dialup modem call progress activity is monitored.

Monitor Dialup Modem Activity to Display (DIAL.DISP) – Determines whether dialup modem call progress activity is monitored to the alphanumeric LED display.

TRACE Enable (TRACE) – Enables/disables the XXXCODE execution trace functions.

Output Devices (OUTPT.DVC)

Standard Output Device (STD.OUT) – Determines where data, such as activity monitoring (previous section) destined for the standard output device is sent. The standard output device can be assigned one of the following values:

Table 4-4 – Standard Output Device Values

NONE	**Null**
COM1	Maintenance port

Display Settings (DISPLAY)

Display Type (-----.DISP) – Determines whether or not the large, full-size display is being used. Value range: Yes/No.

Scroll Delay (SCRLL.DLY) – Expressed as a time delay, in 10 ms increments between each character, determines the speed at which data is scrolled on the alphanumeric display. Value range: 0 – 100. Default: 18.

Refresh Delay (RFRSH.DLY) – Expressed as a time delay, having 10 ms increments between refreshes, determines the alphanumeric display refresh rate (such as when using the display to examine a value). A smaller value causes more responsiveness, but at the expense of other running tasks (e.g., XXXCODE interpreter, telemetry). Value range: 0 – 1000. Default: 50 (0.5 seconds).

Flash Delay (FLASH.DLY) – Expressed as a time delay, having 10 ms increments between states, determines the cycle rate for both alphanumeric display flashing text and flashing alarm LEDs. (It has no effect on flashing markers and LED bar graph over/under-range indicators.) Value range: 0 – 1000. Default: 50 (0.5 seconds).

Tag Display Rate (TAG.RATE) – Expressed as a time delay, having 10 ms increments between states, determines how often the tag name appears on the alphanumeric display when an I/O register (e.g., analog input, analog output) is being examined. For example, the display occasionally flashes L17 when level 17 (AI 17) is being scrutinized. Value range: 0 – 65535. Default: 2000 (20 seconds).

Tag Display Delay (TAG.DLY) – Determines how long each tag string remains on the display. It is given in 10 ms increments. Value range: 50 – 800. Default: 100 (1 second).

Tag Display Control (TAG.CTRL) – Determines if the alphanumeric display shows tag name data. Default: YES.

Default Display Precision (DFLT.PREC) – Determines how many digits of precision are displayed on the alphanumeric display for floating point numbers. Value range: 0 – 7. Default: 1.

XMT/RCV LED Settings (XMT.RCV) – Determines which port(s) cause the XMT and RCV LEDs to illuminate when messages are transmitted and received, respectively. Possible settings are NONE, ALL or any combination of P1, P2, P3, P4. Default: ALL.

LED Display Blanking (LED.BLANK) – Enables/disables blanking of the optional LED display. Time before blanking is defined by MISC > LOCK.TIME. Default (no blanking): NO.

Miscellaneous Settings (MISC)

Write-Protected RAM CRC Rate (WP.CRC.RTE) – Expressed in 10 ms increments, determines how often a CRC (cyclical redundancy check) is performed on write-protected RAM to insure its contents have not become corrupted. The default performs a complete CRC as a background task over the course of each 60 seconds. If set as a very small value, the -------- does its best to complete this task in the allotted time without monopolizing the CPU. A value of 0 disables CRC checking. Value range: 0 – 65535. Default: 6000 (60 seconds).

Keypad Auto-Lock Time (LOCK.TIME) – Expressed in 1 second increments, determines how long the -------- waits from the last keypress to when the keypad is automatically locked and the display is blanked (if LED.BLANK is enabled). Once locked, the operator must reenter the keyword to unlock it. Value range: 0 – 300, where 0 disables auto-locking.
Default: 300 (5 minutes).

Maximum Idle Time (MAX.IDL) – Expressed in milliseconds, used for serial communication ports to allow changing the inter-character gap when receiving bytes from a device. Usually a digital radio with buffering may not transmit the complete protocol message in a single block.
Default: 8.

Network Port Settings (NET.COM1, NET.COM2, NET.COM3, NET.COM4)

Baud (n.BAUD) – Determines the baud rate at which a port communicates. The device being communicated with must also use the same baud rate. Supported baud rates are: 300, 600, 1200, 2400, 4800, 9600, 19200, 38400, 57600.

Table 4-5 – Valid Baud Rates

Port	Description	Baud Rates
NET.COM1	RS-232, no handshake	300 – 57,600 bps
NET.COM2	RS-232, full handshake	300 – 57,600 bps
NET.COM3	RS-232, full handshake	300 – 57,600 bps
NET.COM4	Ethernet	10/100 Mbps
NET.COM6	Modbus® TCP	10/100 Mbps

Notes: There is no NET.COM5.

Any -------- port can be set to any valid baud rate and need not correspond to the rate of any other port.

COM1 defaults to 19,200 bps, 8 bits, no parity on power up. This ensures that communication can be re-established if an incorrect COM1 setting is sent to the --------.

Parity (n.PARITY) – Determines the parity the port uses for communications. The device being communicated with must use the same parity setting. Supported parity codes are:

Table 4-6 – Parity Values

NONE	No parity
ODD	Odd parity
EVEN	Even parity

Stop Bits (n.STOPBIT) – Determines the number of stop bits the port uses for communications. The device being communicated with must use the same number of stop bits. Values: 1, 2.

DCD Function (n.DCD.BUSY) – Determines if the port uses data carrier detect to indicate the line is busy. Multipoint networks use DCD as a busy indicator, whereas point-to-point networks might use DCD to indicate that the carrier is established and it is clear to send.

Value range: Yes/No.

Table 4-7 – Interpretation of Handshake Signals

Port	DCD	CTS	DSR
RS-232 full handshake	Must be asserted for message XMT	Must be asserted for byte XMT	Must be asserted for message XMT
RS-232 no handshake	n/a		

Protocol (n.PROTO) – Determines the basic telemetry protocol is used on this port. Any subdivisions or differences within this protocol are specified in a network connection table entry for the unit to which a message is being sent.

Default value: -----.

Table 4-8 – Telemetry Protocol Values

Field Value	Description
---	--------/---- (----------) or ------- (----------)
MBUS.MSTR	-------- acts as a Modbus® master
MBUS.SLAV	-------- acts as a Modbus® slave
MBUS.EXT	Allows the serial port to act as both Modbus® master and slave A serial port can have both Modbus® slave and a poll/ quiescent tables
DGH.MSTR	-------- acts as a DGH master

Note: For port configuration purposes, Data Express and Data Express Plus is considered one protocol. Thus, multiple --------, ---------, ------, and --------- units can coexist on the same communications link. When a -------- originates a message to a ------ or --------, it distinguishes between them using the Device Type field in its network connection table.

Answer Delay (n.ANSR.DLY) – Expressed in 10 ms increments, controls the time from when a message is received/decoded to when the -------- attempts to answer back—or acknowledge—the message. This delay does not apply to messages originated by the ---------—only replies. Its purpose is to give the other node time to release the communications line.

Value range: 0 – 1000 (0 – 10 seconds).

Leading Pad (n.LEAD.PAD) – Add extra characters to the front of a message for those peripheral communication devices requiring extra time to prepare for an incoming telemetry stream.

Value range: 0 – 20.

Trailing Pad (n.END.PAD) – Add extra characters to the end of the message for those peripheral communication devices requiring extra time to properly end a telemetry stream.

Value range: 0 – 20.

Acknowledge Wait Time (n.ACK.WAIT) – Expressed in 10 ms increments, sets the duration the -------- waits for an acknowledge/response to an outgoing message. The -------- resends the message if no acknowledgment is received prior to n.ACK.WAIT expiration.

Value range: 0 – 1000.

Acknowledge Wait Maximum Count (n.ACK.MAX) – Sets the number of times the -------- waits for an acknowledge/response to an outgoing message before declaring a communications failure. Value range: 1 – 65535.

Intermediate Acknowledge Wait Time (IACK.WAIT) – Expressed in 10 ms increments, sets the duration the -------- waits for an acknowledge/response to an outgoing routed message. The -------- resends the message if no acknowledgment is received prior to IACK.WAIT expiration. Value range: 0 – 1000.

Intermediate Acknowledge Wait Maximum Count (IACK.MAX) – Sets the number of times the -------- waits for an intermediate acknowledge/response to an outgoing routed message before declaring a communications failure. Value range: 1 – 65535.

Table 4-9 – Conditions Considered Busy

Media Type	Condition Treated as Busy
RS-232 full handshake	DSR not asserted DCD according to DCD.BUSY setting
RS-232 no handshake	n/a

Busy Wait Time (n.BSY.WAIT) – Expressed in 10 ms increments, sets the duration the -------- waits before reattempting to send a message when it has encountered a busy condition. Value range: 1 – 4000000.

Busy Wait Maximum Count (n.BSY.MAX) – When it senses a busy condition, sets the number of times the -------- tries to resend a message before declaring a communications failure. Value range: 1 – 65535.

Reconnect Wait Time (n.RCN.WAIT) – Expressed in 10 ms increments, determines how long the -------- waits between attempts to contact a unit for which it has declared a communications failure. Value range: 1 – 4000000.

Transmit Queue Size (n.Q.SIZE) – Determines the number of messages that can be queued, awaiting transmission for this port. Queue size is a function of the media to which the port is connected, its baud rate, lead/trail delay settings, the amount of quiescent messaging, etc. The minimum value must be the number of network connection table entries for this port, plus two (2). This allows room for the -------- to keep a probe message on the queue for each network node with which it is trying to communicate, such that it can reconnect in the event of a failure. The default should suffice, but this value can be increased if an overflow occurs (a log message is generated). Note that a larger queue size takes more system memory. Value range: 1 – 100.

Default: 10.

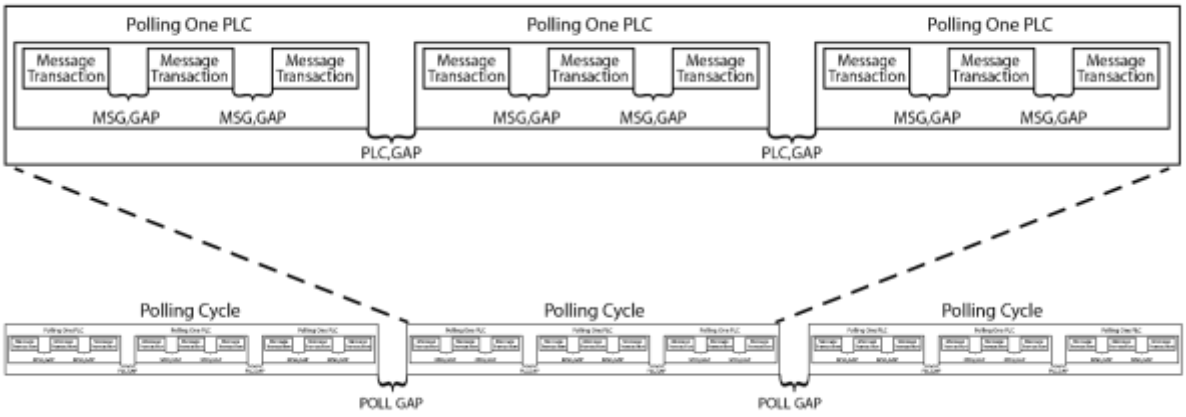
Communications Fail Path ID (n.CM.FL.ID) – Identifies the unit (by path ID) that should be notified in the event of a communications failure. A corresponding entry with the given path ID must exist in the network connection table. Value range: 0 – 65535.

Default: 0 (no unit should be notified).

Poll Delay (n.POLL.DLY) – Expressed in 10 ms increments, specifies a per-port poll delay to indicate how long after power up, or after entering run mode, the -------- should wait to initiate port polling. Value range: 0 – 65535.

Default: 500 (5 seconds).

Figure 4-1 – Polling Cycle



Poll Gap (n.POLL.GAP) – Expressed in 10 ms increments, specifies the delay a -------- should use between each polling cycle (figure 4-1). Value range: 0 – 65535. Default: 100 (1 second).

PLCGap (n.PLC.GAP) – Expressed in 10 ms increments, specifies the delay the -------- uses between polling each successive -------- in a polling cycle (figure 4-1). Value range: 0 – 65535. Default: 0.

Message Gap (n.MSG.GAP) – Expressed in 10 ms increments, specifies the delay the -------- uses between each successive polling transaction with a given -------- (figure 4-1). Value range: 0 – 65535. Default: 0.

Disable DI (n.DISBL.DI) – Turns OFF (disables) the corresponding port such that all telemetry is disabled.

Default values:

2. DISBL.DI	DI 982
3. DISBL.DI	DI 983
4. DISBL.DI	DI 984
6. DISBL.DI	DI 886

Recommended Settings for Timing Parameters

Ideal values for the timing parameters listed above vary by the type of media to which a port is connected, and the network node characteristics the -------- uses to communicate. The settings in Table 4-10 offer good starting points and should yield successful communications in most cases. Occasionally you may need a different value for proper communications or for faster throughput.

All listed timing parameters are in 10 ms time units.

Table 4-10 – Recommended Timing Settings

Feature Code	Common Settings	
RS-232 and RS-485	LEAD.PAD **END.PAD** ACK.WAIT **ACK.MAX** BSY.WAIT **RCN.WAIT**	1 **1** 30 + Dm **10** 10 **6000 (1 min.)**

Dm (delay for maximum message) = 255000/baud rate

Unique Settings	RS-232	RS-485
ANSR.DLY	0	4
BSY.MAX	10	100

Network Port Settings (NET.COM6) Virtual Modbus Port

NET.COM6 is a virtual Modbus® communications port. The properties only apply to the Modbus® TCP protocol on NET.COM4.

Data Express, Data Express Plus, and Modbus® TCP can all coexist on Ethernet port NET.COM4 and NET.COM6.

TCP/IP Ethernet

These NET.COM4 settings specify the following:

4.IP – The -------- IP address.

4.SUBNET – The -------- subnet mask.

4.GATEWAY – The IP address of the device that handles off-network traffic.

4.MAC – The -------- read-only MAC address.

4.APP.PORT – The TCP communications port. Default: 2000

4.ENABLE – If set to OFF, shuts down Modbus® TCP communications. Default: ON

Configuration Table File

You can store the configuration table as an ASCII text file and use -------- 64 upload it to—or download it from—the --------. The file consists of a list of fields and associated values. The field names are the same as the prompt names given in the previous section. The easiest way to create a new configuration table is to upload one from an -------- and then change the fields as desired. The table may then be downloaded to the --------.

Below is a partial listing of an example configuration table source file:

```
: -----[ --------.INFO ]-----
GWAY.NAME=--------
: -----[ DO/DI ]-----
PHYS.DO=32
PHYS.DI=24
INTRN.DI=376
INT.DI.IDX=300
: -----[ AI/AO/SP ]-----
PHYS.AI=8
INTRN.AI=120
PHYS.AO=4
INTRN.AO=116
SP=128
INDEX=128
: -----[ INPT.SCAN ]-----
SCAN.TOP=OFF
SCAN.LINE=0
SCAN.RATE=1000
AI.COUNT=0
```

4.3 XXXCODE Program

XXXCODE program syntax is described in *Chapter 5 – Programming*. The XXXCODE control program is stored within the program source file.

A program file consists of a series of XXXCODE lines in ASCII form. The following is a partial example:

```
10- ACTION T59 ON AND P17 OFF       :This is a comment
20- ACTION T60 INCR                 :This is another comment
100- L20 < SP 20
    ACTION P1 ON
    ELSE
    ACTION P1 OFF
110- L21 < SP21
    ACTION P2 ON
    ELSE
    ACTION P2 OFF
```

Refer to *Chapter 5 – Programming* for more.

4.4 Network Connection Table

For the -------- to originate a message for any network node, a corresponding entry must exist in its network connection table. Here the -------- store and forward routing feature is defined and configured.

The network connection table is a single-column ASCII file (shown here as two columns to conserve space). Its entries have the following format:

```
BEGIN                              SEND.DEV.TYPE=<value>
PATH.ID=<value>                    SEND.MAX.HOP=<value>
ALT.PATH=<value>                   MDM.DISCONNECT=<value>
FROM.NET.ID=<value>                MDM.INIT=<value>
FROM.NODE.ADR=<value>              MDM.DIAL=<value>
DEST.NET.ID=<value>                COMM.FLAG=<value>
DEST.NODE.ADR=<value>              COMM.PCNT=<value>
VIA.NET.ID=<value>                 POLL.REG=<value>
VIA.NODE.ADR=<value>               DEST.IP=<value>
SEND.PORT=<value>
```

Each entry must start with the keyword *BEGIN*.

PATH.ID – A unique identifier that is referenced in quiescent table and polling table entries as a message destination. If a network node originates a message to the -------- and there is no corresponding network connection table entry, the -------- automatically creates one. Such path IDs start at `60000`.

ALT.PATH – Identifies an alternate path to be used if communication fails on the `PATH.ID`.

FROM.NET.ID and *FROM.NODE.ADR* – For incoming messages using the (legacy) ----- message protocol. Identifies a source address of an incoming message to capture and reroute. Both values should be set to `0.0` unless you are actively capturing and forwarding incoming ----- messages.

DEST.NET.ID and *DEST.NODE.ADR* must match the incoming message for subsequent forwarding.

DEST.NET.ID and *DEST.NODE.ADR* – Identify the final destination node to which messages should be addressed when using this path. If a ---------- (---------), DGH, or Modbus protocol is used, set *NET.ID* field to `0`, since these protocols do not use it.

VIA.NET.ID and *VIA.NODE.ADR* – Identifies a forwarding address through which to route a message so it can reach its final destination, as identified in *DEST.NET.ID* and *DEST.NODE.ADR*.

SEND.PORT – Sets the communications port to use to transmit messages.

SEND.DEV.TYPE – Defines the forwarding or destination node device type. For protocols other than ---, set this value to *NONE*.

If --- is set as the protocol in the configuration table, then this value must be -----, ------, or -------- to distinguish between message types. The ------ device type transmits a message with an envelope supporting store and forward message routing.

SEND.MAX.HOP – Determines the number of forwards a message can take before it is declared an orphan and killed. This value is only supported by the ------- protocol.

MDM.DISCONNECT – Applies only to dialup communications and identifies the disconnect sequence needed for the forwarding or destination node. This field has two possible values:

- *MASTER* – After a dialup connection is established, do not terminate the connection. Wait for the other end to terminate, because it may have data for which it needs to poll and can take advantage of the existing connection. It is then up to the master to terminate the connection.
- *PEER* – Terminate the dialup connection after the message(s) have been delivered.

MDM.INIT – Applies only to dialup connections. It is a character string to be sent to initialize a modem on this port. Currently this field is not implemented and should be left blank.

MDM.DIAL – Applies only to dialup connections. It is a character string to dial and create a modem connection with a node. Currently, the only supported string consists of ATDT followed by a phone number, e.g., `ATDT395-8800` (dashes are allowed, but not spaces).

COMM.FLAG – Specifies a digital input (DI) or output (DO) to track the health of the communications path. If a communications failure occurs on the path, the DI/DO is set to `ON`. If communication is restored, the DI/DO is cleared to the `OFF` state.

The DI/DO can be designated as an alarm in the LED table and will have normal alarm functionality.

Acceptable values are `NONE`, `DI`*nn*, or `DO`*nn*, (where *nn* is a valid DI or DO register index).

Default: `NONE`

COMM.PCNT – Specifies a register to track the successful communications percentage on this path. This value reflects the success of transactions *originated* from this -------- path. It is computed by tracking the last 32 originated transactions (for instance, 30 successful out of 32 = 94%).

A different register should be used for each path.

Incoming transactions are not included, for the -------- cannot determine if the originator has received the --------'s response.

If a communications failure occurs (no response after designated number of retries), the percentage is immediately set to `0`.

Permitted values are `AI`*nn* or `AO`*nn*, (where *nn* is a valid analog input (AI) or output (AO) register index). Default: `NONE`

POLL.REG – Specifies an analog output (AO) register containing the priority of polling the designated device receives relative to other devices.

Value

`0` – Disable polling for this device (useful to temporarily disable device polling without removing it from the poll table).

`1` – Poll the device every poll cycle (see *Chapter 6 – Networking & Telemetry*).

`2` – Poll the device every other time, etc.

`NONE` – Poll priority defaults to 1.

Default: `NONE`

DEST.IP – Specifies the destination TCP/IP network address.

SAMPLE—Below is a sample network connection table file (partial, with comments):

```
                        : See routing setup for further examples
BEGIN                   : All entries have a BEGIN
PATH.ID=1               : Uniquely identifies entry, reference in
                        :    quiescent table and or polling table
ALT.PATH=0              : PATH.ID of another entry for alternate
                        :    path (0 = none) used on comm fail escalation
FROM.NET.ID=0           : Capture and forward any received message
FROM.NODE.ADR=0         :    that matches this source address and matches
                        :    the DEST.NET.ID and DEST.NODE.ADR fields.
                        : Allows routing of standard ---- messages
                        :    0.0 disables this feature
DEST.NET.ID=1           : Network ID of final destination unit
DEST.NODE.ADR=2         : Node address of final destination unit
VIA.NET.ID=0            : To reach the final destination, route the
VIA.NODE.ADR=0          :    message through this address
                        :    0.0 disables this feature
SEND.PORT=3             : Communications port to use
SEND.DEV.TYPE=----      : Device type of unit
SEND.HOP.MAX=0          : No routing needed
MDM.DISCONNECT=PEER     : Modem hang-up sequence (Master or Peer)
MDM.INIT=               : Modem initialization string
MDM.DIAL=ATDT395-8800   : Modem dial string
COMM.FLAG=DI300         : DO/DI to track communications failure/success
COMM.PCNT=AO200         : AI/AO to track communications success rate
POLL.REG=NONE           : AO register, default NONE is a polling priority of 1
DEST.IP=192.168.10.10   : If on Ethernet, then assign TCP/IP network address
```

4.4.1 Store and Forward Message Routing Setup

Add the appropriate entries to the network connection table to route store and forward messages.

4.4.2 Store and Forward—Data Express Plus and Data Express

<company> has developed two messaging protocols; --------- is for the --------/---------- models, whereas the ------ uses --------. In the following example (consisting of four entries and continuing on the next two pages), the --------/--------- forwards native messages from serial to Ethernet and Ethernet back to serial.

Here the polling unit address 1.255 is polling over serial port two (P2) to address 1.3:

SCADA or Polling Unit TLM 1.255

```
BEGIN
PATH.ID=102
ALT.PATH=0
FROM.NET.ID=0
FROM.NODE.ADR=0
DEST.NET.ID=1          : first part of dest addr 1.3
DEST.NODE.ADR=3        : second part of dest addr 1.3
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=2            : send poll msg out P2
SEND.DEV.TYPE=----
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=0.0.0.0
```

The NetConnect table in address 1.2 (below) defines the store and forward operation. Here the message is being trapped and forwarded over Ethernet.

Store and Forward Unit TLM 1.2 to TLM 1.3

```
BEGIN
PATH.ID=104
ALT.PATH=0
FROM.NET.ID=1             : msg received from 1.255
FROM.NODE.ADR=255         : msg received from 1.255
DEST.NET.ID=1
DEST.NODE.ADR=3           : for dest addr 1.3
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=4               : forward msg out Ethernet
SEND.DEV.TYPE=-----
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=192.168.15.103    : IP address
```

Store and Forward Unit TLM 1.2 Back to TLM 1.255

```
BEGIN
PATH.ID=102
ALT.PATH=0
FROM.NET.ID=1
FROM.NODE.ADR=3        : msg received from 1.3
DEST.NET.ID=1
DEST.NODE.ADR=255      : for destination 1.255
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=2            : forward msg out P2
SEND.DEV.TYPE=----
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=0.0.0.0
```

TLM 1.3 Back to TLM 1.255

```
BEGIN
PATH.ID=104
ALT.PATH=0
FROM.NET.ID=0
FROM.NODE.ADR=0
DEST.NET.ID=1
DEST.NODE.ADR=255
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=4
SEND.DEV.TYPE=----
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=192.168.15.102
```

4.4.3 Store and Forward—Modbus®

In the following example (consisting of four entries and continuing on the next two pages) the --------/-------- forwards Modbus® messages from serial to Ethernet and Ethernet back to serial.

Here the polling unit address 0.255 is polling over serial port two (P2) to address 0.3.

SCADA or Polling Unit TLM 0.255

```
BEGIN
PATH.ID=102
ALT.PATH=0
FROM.NET.ID=0
FROM.NODE.ADR=0
DEST.NET.ID=0          : first part of dest addr 0.3
DEST.NODE.ADR=3        : second part of dest addr 0.3
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=2            : send poll msg out P2
SEND.DEV.TYPE=NONE
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=0.0.0.0
```

The NetConnect table in address 0.2 (below) defines the store and forward operation. Here the message is being trapped and forwarded over Ethernet.

Store and Forward Unit TLM 0.2 to TLM 0.3

```
BEGIN
PATH.ID=104
ALT.PATH=0
FROM.NET.ID=0            : msg received from 0.255
FROM.NODE.ADR=255        : msg received from 0.255
DEST.NET.ID=0
DEST.NODE.ADR=3          : for dest addr 0.3
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=4              : forward msg out Ethernet
SEND.DEV.TYPE=NONE
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=192.168.15.103   : IP address
```

Store and Forward Unit TLM 0.2 Back to TLM 0.255

```
BEGIN
PATH.ID=102
ALT.PATH=0
FROM.NET.ID=0
FROM.NODE.ADR=3        : msg received from 0.3
DEST.NET.ID=0
DEST.NODE.ADR=255      : for destination 0.255
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=2            :forward msg out P2
SEND.DEV.TYPE=NONE
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=0.0.0.0
```

TLM 0.3 Back to TLM 0.255

```
BEGIN
PATH.ID=104
ALT.PATH=0
FROM.NET.ID=0
FROM.NODE.ADR=0
DEST.NET.ID=0
DEST.NODE.ADR=255
VIA.NET.ID=0
VIA.NODE.ADR=0
SEND.PORT=4
SEND.DEV.TYPE=NONE
SEND.MAX.HOP=1
MDM.DISCONNECT=PEER
MDM.INIT=
MDM.DIAL=
COMM.FLAG=NONE
COMM.PCNT=NONE
POLL.REG=NONE
DEST.IP=192.168.15.102
```

4.5 Quiescent Table

The quiescent table determines which data points are sent using quiescent telemetry, and under which conditions. The quiescent table consists of a list of entries in any order. There are two types of entries: *change*-based and *interval*-based.

4.5.1 Change-Based Entries

The basic principle of change-based quiescent telemetry is that whenever a source register changes by more than a threshold register value, a source register value is sent to a destination register. The amount of change is determined by the last value that was sent.

Digital inputs and outputs follow this principle, except that there is no threshold register since a DI/DO has one of only two values (`0`/`1`). Transmission of timers/counters occurs when an associated DI/DO threshold register changes state; this does not depend on the timer/counter value. Additional details/restrictions depend upon the telemetry protocol being used, as shown in the following tables.

4.5.2 Interval-Based Entries

Interval-based entries are fairly simple. Here a source register value can be sent to any destination unit register at periodic intervals. The interval is specified in 10ms time units (as are all telemetry timing parameters). A `0` (zero) value means as fast as possible. Actual times may vary depending on system load and network traffic.

Based on the protocol used, table 4-11 summarizes valid quiescent table entry combinations. The threshold register column applies only to change-based entries.

Table 4-11 – Quiescent Table Combinations— ---------- Protocol

Source Register	Allowed Destination Registers	Allowed Threshold Registers
AI	AI, AO, SP	AI, AO, SP
AO		
SP		
T	T	DI, DO
DI	DI, DO	N/A
DO		

Table 4-12 – Quiescent Table Combinations— ---------- Protocol

Source Register	Allowed Destination Registers	Allowed Threshold Registers
AI	AI, AO, SP	AI, AO, SP
AO		
SP		
T	T	DI, DO
IR	IR, AO	SP, IR
DI	DI, DO	N/A
DO		

Table 4-13 – Valid Quiescent Table Entries

Modbus Destination Register	-------- Source Register
40001…41999F Holding	AO0…AO*nn*,AI0…AI*nn*
00001…01999 Coil	DI0…DI*nn*,DO0…DO*nn*

The quiescent table is an ASCII file and has entries using the following format:

```
BEGIN
PATH.ID=<value>
INTERVAL=<value>
SRC.REG=<value>
DST.REG=<value>
THRES.REG=<value>
TRIG.REG=<value>
```

Each entry must start with the keyword *BEGIN*.

PATH.ID – Identifies the path to send the data point to its destination. To be valid, it must correspond to a path defined in the network connection table.

INTERVAL – Specifies the transmission interval for interval-based entries and is expressed in 10 ms units (e.g., `3000` equates a 30-second interval). A `0` (zero) value means the entry is change-based.

SRC.REG – Identifies the source register; its value should be read and sent (e.g., `AO99`).

DST.REG – Identifies the destination register (where the value is to be stored in the destination unit).

If the -------- is a Modbus master, then the Modbus register is entered as:

- `40001` for the first unsigned integer holding register
- `40001F` if receiving floating point
- `40001FW` to receive floating point and swap 16-bit words
- `40001S` if receiving a signed integer

The optional ^ (carat) denotes a scaling factor. For example, ^1 means divide by 10 after receiving the value; `^2` means divide by 100.

THRES.REG – Set to `NONE` for interval-based entries. If `INTERVAL` is set to `0` (above), identifies the threshold register for change-based entries.

TRIG.REG – Specifies an optional trigger register that exerts control over the associated quiescent data point transmission. Permitted values are `NONE` or a digital input (DI) register `DI`*nn*.

If a register is specified, it has the following effect:

Table 4-14 –Trigger Register Values

Condition	Effect
TRIG.REG = `0`	The entry is NEVER sent quiescently
TRIG.REG = `1`	The entry is sent as normal, depending on interval or threshold
TRIG.REG transition `0` to `1`	The quiescent entry is sent immediately, regardless of interval or threshold values

For the quiescent task to detect a `0` to `1` transition, both the `0` and `1` states should be held for at least the quiescent scan time duration (set in the configuration table). You can use the same trigger register for multiple quiescent entries.

SAMPLE—Below is a partial listing of a sample quiescent telemetry table file:

```
: Comment line
: Send AO5 to 40005 as a float through path 1,
: every 3000 x 10 ms (30 seconds)
BEGIN             : Each entry starts with a BEGIN
PATH.ID=1         : Where to send it
INTERVAL=3000     : How often to send
SRC.REG=AO5       : Where to get value from
DST.REG=40005F    : Modbus holding register 5; send as Float
THRES.REG=NONE    : There is no threshold for interval-based
TRIG.REG=NONE     : No trigger Register specified

BEGIN
PATH.ID=1         : Send AI7 to AO10 through path 1
INTERVAL=0        : 0 means change-based
SRC.REG=AI7
DST.REG=AO10
THRES.REG=AO17    : Change threshold is stored in AO17
TRIG.REG=NONE     : No trigger Register specified
```

4.6 Polling Table

Referred to as polling, an -------- can retrieve register values from remote units and then store responses in local registers. The -------- can poll other -------- controllers, ------- controllers, --------- controllers, Modbus® slave devices, and DGH products.

An ASCII polling table partially defines polling functions. It has entries in the following format:

```
BEGIN
PATH.ID=<value>
PRIORITY=<value>
SRC.REG=<value>
DST.REG=<value>
```

Other tables contain additional fields that control polling operation:

- Configuration table – POLL.DLY, POLL.GAP, --------.GAP, MSG.GAP
- Network connection table – POLL.PRI

See the configuration table (section 4.2) and network connection table (section 4.4) for more details.

4.6.1 Polling Table Format

Each entry must start with the keyword *BEGIN*.

PATH.ID – Identifies the unit to be polled. There must be a corresponding network connection table entry (refer to section 4.4).

PRIORITY – Specifies the register polling priority relative to other entries for the same PATH.ID. The permitted range is `0-65535`.

`0` – Disable polling (useful for temporarily disabling polling of a register without removing it from the table).

`1` – Poll for this register every time the -------- identified by this PATH.ID is polled.

`2` – Poll every other time.

Default: `1`

SRC.REG – Identifies the register to retrieve information from in the remote unit.

If the -------- is a Modbus® master then the Modbus register is entered as:

- `40001` for the first unsigned integer holding register
- `40001F` if receiving floating point
- `40001FW` to receive floating point and swap 16-bit words
- `40001S` if receiving a signed integer

The optional ^ (carat) denotes a scaling factor. For example, ^1 means divide by 10 after receiving the value; ^2 means divide by 100.

DST.REG – Identifies the local -------- register in which to store the retrieved value.

SAMPLE—Below is a partial listing of a sample polling table file:

```
: Comment line
: Poll for Modbus 40001 float, PATH ID 1, store in AO1 every poll cycle.
BEGIN            : Each entry starts with a BEGIN
PATH.ID=1        : Where to obtain data
PRIORITY=1       : How often to poll
SRC.REG=40001FW  : Receive holding reg 1 as floating point value and swap words
DST.REG=AO1      : Store value to AO1

: Poll for Modbus 40001 signed int, divide by 100, store in AO1 every cycle.
BEGIN            : Each entry starts with a BEGIN
PATH.ID=1        : Where to obtain data
PRIORITY=1       : How often to poll
SRC.REG=40001S^2 : Receive holding reg 1 as a signed int and divide value by 100
DST.REG=AO1      : Store value to AO1

: Poll for Modbus 40001 unsigned int, divide by 100, store in AO1 every cycle.
BEGIN            : Each entry starts with a BEGIN
PATH.ID=1        : Where to obtain data
PRIORITY=1       : How often to poll
SRC.REG=40001^2  : Receive holding reg 1 as unsigned int and divide value by 100
DST.REG=AO1      : Store value to AO1
```

The following tables show the permitted polling register types for each telemetry protocol.

*Table 4-15 – Polling Registers— ---------- **Protocol***

Source Register	Permitted Destination Registers
AI	AI AO
AO	
SP	
T	T
DI	DI DO
DO	

*Table 4-16 – Polling Registers— ---------- **Protocol***

Source Register	Permitted Destination Registers
AI	AI AO
AO	
SP	
T	T
IR	IR
DI	DI DO

Table 4-17 – Valid Polling Table Entries

Modbus Source Register	-------- Destination Register
40001…41999F Holding	AO0…AO*nn*,AI0…AI*nn*
10001…11999 Status	DI0…DI*nn*,DO0…DO*nn*
30001…31999 Input	AO0…AO*nn*,AI0…AI*nn*

Two sample polling table entries are given below:

```
: Comment line
: Poll for AI 0, PATH ID 1, and store in AI0 every poll cycle.
BEGIN         : Each entry starts with a BEGIN
PATH.ID=1     : Where to obtain data
PRIORITY=1    : How often to poll
SRC.REG=AI0   : Which register to get
DST.REG=AO20  : Which register to store value in

: Poll for AO 12, PATH ID 3, and store in AI 14 every other poll cycle.
BEGIN
PATH.ID=3
PRIORITY=2
SRC.REG=AO12
DST.REG=AO14
```

4.7 Setpoint Table

You can define a list of setpoints and associated values that can be downloaded to the -------- in the (ASCII) setpoint table.

SAMPLE—Below is a partial listing of a sample setpoint table file:

```
SP0 = 17.01
SP1 = 93.0
SP2 = 574
SP3 = 13.33
SP4 = 9.75
SP5 = 0
SP6 = 10
SP7 = 6784.455
SP8 = 4995.33
SP9 = 234.0
SP10 = 432.9923
SP11 = 3.2415
SP12 = 2.7818
SP13 = 3600.0
SP14 = 24.0
```

4.8 Filter Table

You can define a list of analog input filter constants that can be downloaded to the -------- in the (ASCII) filter table.

Note: The AORESET field allows the associated analog output to be set to 0 (zero); this can correspond to 4mA on a transition from run mode to standby mode.

SAMPLE—Below is a partial listing of a sample filter table file:

```
AORESET0=OFF
AORESET1=ON
AIFILTER0=5.0
AIFILTER1=7.5
AIFILTER2=100
AIFILTER3=1.0
AIFILTER4=2.5
AIFILTER5=3.0
AIFILTER6=1.57
```

4.9 Calibration Table

You can define -------- analog input (AI) and output (AO) calibration settings in the (ASCII) calibration table. Normally, calibration is performed using the keypad, then the calibration table can be uploaded from the -------- for safekeeping.

SAMPLE—Below is a partial listing of a sample calibration table file:

```
BEGIN              : Each entry starts with BEGIN
REG=AI0            : Identifies the register (AI or AO)
TYPE=4-20MA        : Identifies the type (4-20MA or 0-5V)
1.ENGR=0.0         : Engineering value for calibration point 1
1.METER=4.0        : Meter reading (mA) for calibration point 1
2.ENGR=100.0       : Engineering value for calibration point 2
2.METER=20.0       : Meter reading (mA) for calibration point 2

BEGIN
REG=AO10
TYPE=0-5V
1.ENGR=15.0
1.METER=0.0
2.ENGR=10000.0
2.METER=5.0
```

4.10 Precalibration Table

The precalibration table lets each AI or AO channel be accurately adjusted so that each responds identically, providing variation compensation between channels and -------- units. Once precalibrated, an AI/AO channel accurately reflects the voltage or current within its allowable range. Precalibration is performed through the keypad, then the resultant ASCII table can be uploaded and saved for later downloading to the --------.

Note: A precalibration table is specific to each --------, so you should download a given table to the respective -------- from which it was generated. A precalibration table is not usually downloaded unless the -------- has been sent an NMI (which erases all memory and resets the device to a factory default condition—this is required if firmware is changed).

Refer to *Chapter 7 – **Error! Reference source not found.*** for more information about the precalibration process.

Each entry must start with the keyword *BEGIN*.

SAMPLE—Below is a partial listing of a sample precalibration table file:

```
BEGIN              : Each entry starts with a BEGIN
REG=AI0            : Register type and number
TYPE=4-20MA        : Range for precalibration
1.RAW=714          : Raw value found at first point (4 MA)
2.RAW=3643         : Raw value found at second point (20 MA)

BEGIN
REG=AI1
TYPE=4-20MA
1.RAW=698
2.RAW=3627

BEGIN
REG=AO0
TYPE=4-20MA
1.RAW=703
2.RAW=3629
```

4.11 Macro Table

You can define entries to define macro key programming in an ASCII macro table.
For more information see section *5.7 – Program Macro Keys*.

SAMPLE—Below is a partial listing of a sample macro table file:

```
1- EXAM L1
2- EXAM T59
3- L3 < SP4 ACTION P17 ON AND T13 OFF ELSE ACTION P17 OFF AND T13 ON
4- ACTION T2 INCR AND EXAM T2
```

4.12 Tag Table

You can use the tag table to associate identification and other data with -------- registers, used for display and other purposes. The tag table is stored in write-protected RAM and can be uploaded/downloaded using ---------- 64 (*Chapter 8.0 – ---------- 64 IDE*).

The (ASCII) tag table is comprised of multiple entries having the following format:

```
BEGIN
REG=<value>
PREC=<value>
UNITS=<value>
TAGNAME=<value>
DESCRIPTION=<value>
```

Each entry must start with the keyword *BEGIN*.

REG – Identifies the -------- register for which an entry is being made. Permitted values are AI (LEVEL), AO (ANAOUT), DO (PUMP), DI (STATUS), SP, IR (INDEX), T (TIMER). The register type is followed by a register index.

PREC – Specifies the precision (number of digits to the right of the decimal point) displayed when the register is shown on the -------- alphanumeric display. Value range: 0 – 7.

UNITS – An optional 4-character field used to specify engineering units associated with a register (e.g., PSI, GPM). The units are displayed on the alphanumeric display using the EXAM function. The unit is shown along with the register value if enough space exits on the display. Otherwise the unit appears only with the register type and index when they is periodically displayed.

TAG.NAME – An optional 60-character field used to hold an abbreviated register identifier (e.g., LIT-100, SYS PRES). It periodically appears on the alphanumeric display when the register is being examined.

DESCRIPTION – This 60-character field holds a longer register identification. It provides the label information for the webserver pages and the color HDMI touchscreen display. It periodically appears on the alphanumeric display when the register is being examined.

When using the EXAM function to examine a register, the following appears on the alphanumeric display in this order:

- Value, units
- Register type and index, units
- Tag name
- Description

The information display is controlled by the TAG.RATE, TAG.DLY, and TAG.CTRL fields in the configuration table. See section *4.2.3.11 – Display Settings* for a description.

When you actively examine a register, pressing the [EXAM] key twice causes any register tag name information to be displayed. If you are not actively examining a register, pressing the [EXAM] key twice shows the last register displayed—but not its tag information. After the delay specified in TAG.RATE, the tag information is periodically displayed.

If actively examining a register and TAG.CTRL = 0 (off), then all tag name strings are displayed when you press [EXAM] twice. However, the tag information is not displayed on a periodic basis.

SAMPLE—Below is a partial listing of a sample tag table file:

```
BEGIN
REG=AO10
PREC=0
UNITS=FT
TAGNAME=PIT-1000
DESCRIPTION=TANK LEVEL

BEGIN
REG=T40
PREC=3
TAGNAME=BS-1234
DESCRIPTION=BACKSPIN DELAY
```

4.13 LED Table

Use the LED table to determine:

- The mapping of DI/DO states to LEDs
- The mapping of bar graph displays to LEDs
- Which DIs/DOs/LEDs are treated as alarms
- Which DO/LED(s) are considered as the common alarm
- Which DOs are to be treated as pulse DOs (see event counter section)
- LED colors and tag mapping to display data on the color HDMI touchscreen and webserver pages

Any displayable DI/DO state can be assigned to any LED annunciator, and any LED can be designated as an alarm. The LED table can be uploaded from and downloaded to the -------- using ---------- 64 (*Chapter 8.0 – ---------- 64 IDE*).

A feature the LED table supports is the assignment of multiple LEDs (as many as desired) to a single DI or DO. They can be assigned in any fashion, subject to the following rules:

- A single LED cannot be assigned to more than one DI/DO.
- An LED cannot be used for both a DI/DO and as part of a bar graph display.
- LEDs must be listed in the LED table in ascending order.
- Phantom DIs (those between the last physical DI and the index defined by the INT.DI.IDX configuration table parameter) cannot be displayed.

Figure 4-2 shows the numbering system used in the table to identify LEDs.

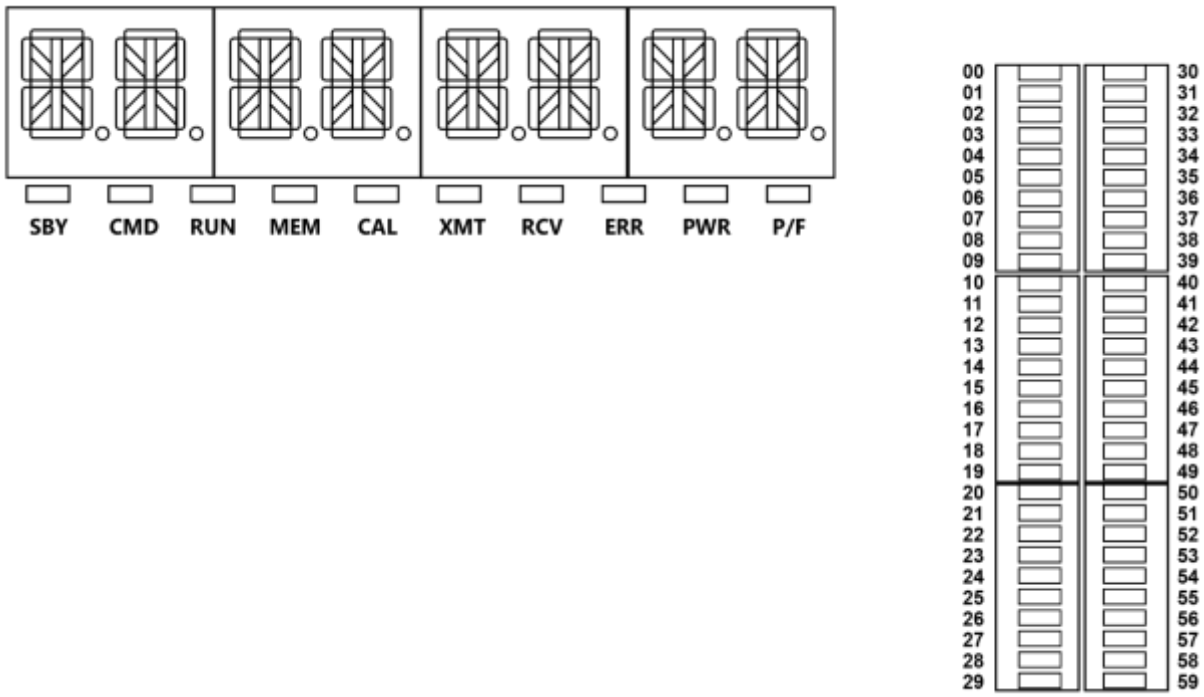


Figure 4-2 – -------- Compact Display LED Numbering

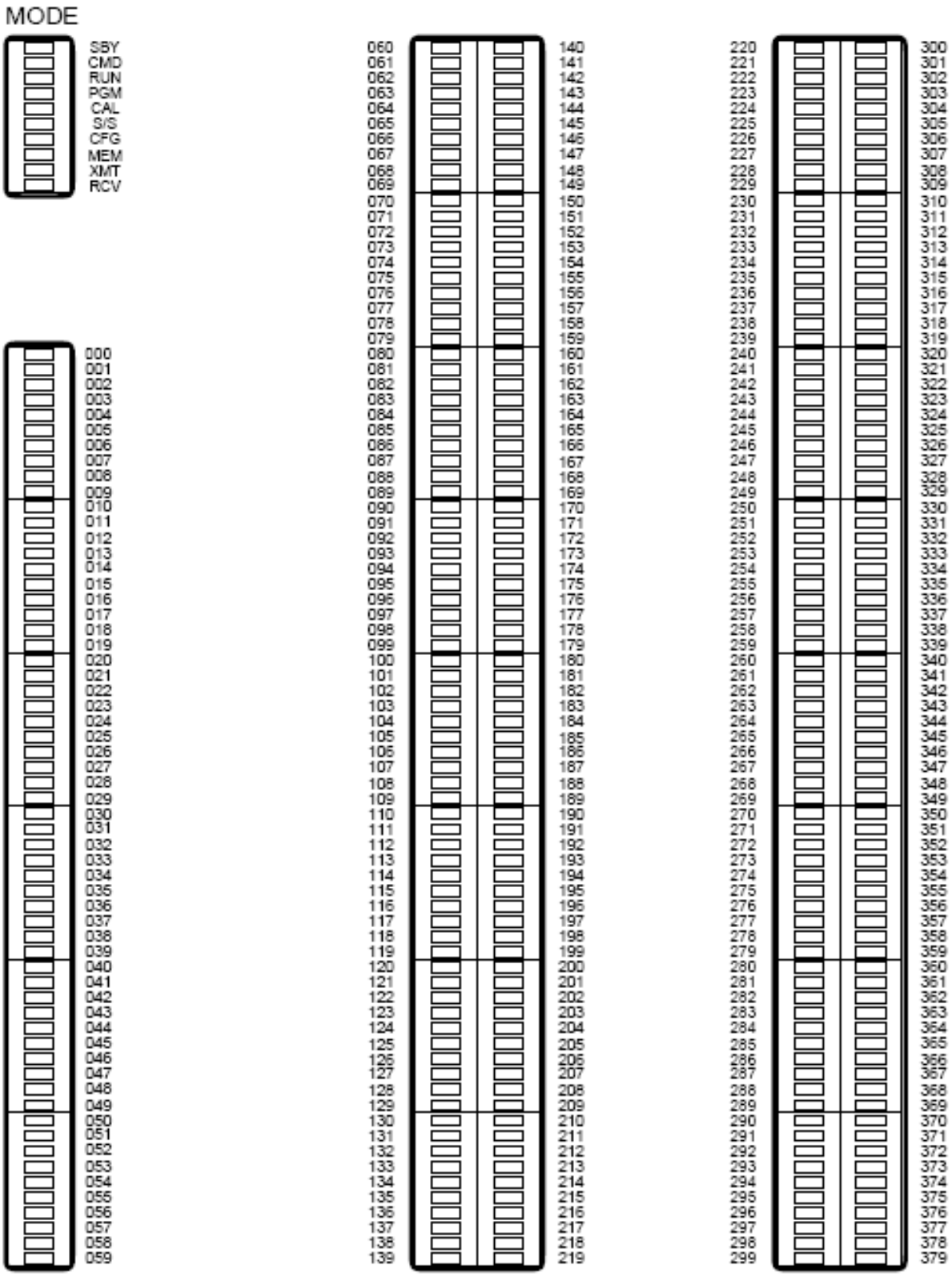


Figure 4-3 – -------- Full-Size Display LED Numbering

Any number of bar graphs can be defined, up to the number that can be displayed without overlapping (38, since each bar graph uses at least 10 LEDs). The bar graph section of the LED table must follow the DI/DO/LED mapping section. The format of a bar graph entry is as follows:

Each entry must start with the keyword *BEGIN*.

REGISTER – Indicates which register the bar graph reflects.

TOP.LED – Use to determine which LED begins the bar graph.

NUM.LEDS – Indicates how many LEDs to use for the bar graph. None of these LEDs can be assigned to anything else.

LOW.VAL – None of the bar graph LEDs will be on when the REGISTER value reaches this value. If the REGISTER value falls below this value then the bottom bar graph LED flashes.

HIGH.VAL – All of the bar graph LEDs will be on when the REGISTER value reaches this value. If the REGISTER value exceeds this value then the top bar graph LED flashes.

LL.MRK.REG (low, low marker register), *L.MRK.REG* (low marker register), *H.MRK.REG* (high marker register), and *HH.MRK.REG* (high, high marker register)—an LED blinks at the bar graph location corresponding to each register value.

Enter NONE, AI, AO, or SP followed by the register index (except for NONE).

LBL<LED number> indicates the label to be used for either the HTML touchscreen display or the webserver pages. This eliminates any special programming for these displays.

COLOR<LED number> – Color to be used for either the HTML touchscreen display or webserver pages. No special programming is required.

Default: Red

SAMPLE—Below is a partial listing of a sample LED table file:

```
LED0=DI9
LBL0=”Pump 1 Call“   : Label for touchscreen and webserver
COLOR0=”GREEN”       : Color label for touchscreen and webserver
LED1=DI9
LBL1=”Pump 2 Call“   : Label for touchscreen and webserver
COLOR1=”GREEN”       : Color label for touchscreen and webserver
LED2=DI10A           : ‘A’ = alarm
LED60=DI0
LED61=DI1A
LED62=DI2
LED140=DO0CF         : ‘C’ = common alarm, ’F’ = flash output
LED141=DO1AF         : Alarm with flashing output
LED141=DO1P          : ‘P’ = pulse DO
LBL1=” “             : Label for touchscreen and webserver
COLOR1=””            : Color label for touchscreen and webserver
```

```
      : The following defines a bar graph display for AI7
BEGIN
REGISTER=AI7          : AI, AO
TOP.LED=60            : Top LED for the bar graph
NUM.LEDS=80           : How many LEDs to use for the bar graph
LOW.VAL=0.0           : Value corresponding to all LEDs off
HIGH.VAL=100.0        : Value corresponding to all LEDs on
LL.MRK.REG=SP1        : NONE, AI, AO, SP
L.MRK.REG=SP2         : NONE, AI, AO, SP
H.MRK.REG=SP3         : NONE, AI, AO, SP
HH.MRK.REG=SP4        : NONE, AI, AO, SP

      : The following defines a bar graph display for AO3
BEGIN
REGISTER=AO3
TOP.LED=220
NUM.LEDS=80
LOW.VAL=50.0
HIGH.VAL=60.0
LL.MRK.REG=NONE
L.MRK.REG=NONE
H.MRK.REG=NONE
HH.MRK.REG=NONE
```

4.14 Archive Array Table

The archive module can log data at 10 ms intervals across all archives; each recorded entry gets a time stamp accurate to 10 ms. Data logs are stored in a user-installed USB flash drive. Every time the ---------- scans an input, or a register write occurs (by XXXCODE, telemetry, or the user), it checks to see if the register is supposed to be archived. This ensures that all data is trapped and processed (unlike other data loggers which have to sample the physical I/O and therefore may or may not synchronize well with the process controller).

The -------- can perform three basic archival types:

- Average/min/max data compression
- Change-based (with associated delta change registers)
- Alarm event

Archive array data integrity is maintained across power failures; archiving is resumed upon power restoration. Use ---------- 64 to configure, upload and download the `filename.arc` table.

The time-stamped data is logged to a USB flash drive inserted into any one of the USB ports. Archive arrays are stored on the flash drive in the following directory:

```
/fs/hd1-usbstick/----------_tables
```

The arrays are stored as .CSV files, with file names designated as follows:

```
PUMP_CALL_P2_100_031715172044.csv
```

Where: *ARRY.NAME_ARRY.NUM_MDYHMS.csv*

There are a number of ways to retrieve the archive arrays, as shown next.

4.14.1 Archive Array Table Setup

The ASCII array table defines the operation of the data archives. Its fields are designed to be easy to use, while also offering maximum control flexibility.

Time units are expressed using the suffix *S* (seconds), *M* (minutes), *H* (hours), or *D* (days); only one unit can be used per field. Time unit fractions are permitted (e.g., `0.5S` for half a second). An archive array table has one or more entries of the following format:

```
BEGIN
INPUT=<value>
ARRY.NUM=<value>
ARRY.TYPE=<value>
LOG.RATE=<value>
ARRY.NAME=<value>
DELTA.REG=<value>
MN.MX.RATE=<value>
ARC.SBY=<value>
ARC.TRIG=<value>
ARC.RESET=<value>
ARC.STATE=<value>
FILE.TIME=<value>
FILE.SIZE=<value>
FILE.MAX=<value>
```

One entry establishes one archiving array, and each must start with the keyword *BEGIN*.

INPUT – Specifies the input register (containing the values to be stored in the archive array).
Permitted values are: AI (analog input), AO (analog output), DI (digital input), DO (digital output), or NONE.

ARRY.NUM – Assigns a number to the archive array for purposes of identification.

ARRY.TYPE – Specifies the archive array type, determining its basic operation.
Permitted values are COMPRESS, CHNG.BASE, or ALARM.SUM.

LOG.RATE – Specifies the archiving interval (how often data is stored),
e.g., `1M` representing one minute.

ARRY.NAME – Stores up to 60 characters to describe the archive array.

DELTA.REG – (Only applicable when ARRY.TYPE = CHNG.BASE and INPUT = AI *or* AO)
The delta change register holds the threshold for archiving data on change.
Default: NONE

MN.MX.RATE – (Only applicable when ARRY.TYPE = COMPRESS)

Specifies the rate at which to archive min/max values. Permitted values are:

NONE (no min/max recording)

LOG.RATE (store min/max values at the same rate as the average)

A specific time period (e.g., 30M for 30 minutes)

Default: LOG.RATE

ARC.SBY – Archiving occurs when the -------- is in standby mode when set to ON.

If set to OFF, archiving only occurs when the -------- is in run mode.

Default: ON

ARC.TRIG – Specifies the trigger register that starts/stops archiving. Values are DI*nn* or NONE.

The trigger register must have an OFF to ON state change for archiving to start.

When the trigger register is OFF, no archiving takes place for the associated array.

Default: NONE

ARC.RESET – Specifies an archive array reset register. Permitted values are DI*nn* or NONE.

Changing the reset register state from OFF to ON causes the associated archive array to reset, where archiving is restarted.

Default: NONE

ARC.STATE – Specifies a register used to show the state of the archive array. Permitted values are DI*nn* or NONE. This register is ON when actively archiving and OFF when archiving is stopped or complete.

Default: NONE

FILE.TIME – Specify (e.g., `30D` for 30 days).

Default: NONE

FILE.SIZE – Specify (e.g., `100000000` for 100M)

Default: NONE

FILE.MAX – Specify (e.g., `10` for 10 files).

Default: NONE

SAMPLE—Below are various examples of a single-column archive array table file (shown here as three columns only to conserve space):

```
BEGIN                   BEGIN                       BEGIN
INPUT=AI0               INPUT=AO10                  INPUT=NONE
ARRY.NUM=100            ARRY.NUM=101                ARRY.NUM=102
ARRY.TYPE=CHNG.BASE     ARRY.TYPE=COMPRESS          ARRY.TYPE=ALARM.SUM
LOG.RATE=NONE           LOG.RATE=1M                 LOG.RATE=NONE
ARRY.NAME=Tank Level    ARRY.NAME=Pump Speed        ARRY.NAME=Alarms
DELTA.REG=SP0           DELTA.REG=NONE              DELTA.REG=NONE
MN.MX.RATE=NONE         MN.MX.RATE=LOG.RATE         MN.MX.RATE=NONE
ARC.SBY=ON              ARC.SBY=OFF                 ARC.SBY=OFF
ARC.TRIG=NONE           ARC.TRIG=DI300              ARC.TRIG=NONE
ARC.RESET=NONE          ARC.RESET=DI301             ARC.RESET=NONE
ARC.STATE=NONE          ARC.STATE=DI302             ARC.STATE=NONE
FILE.TIME=60D           FILE.TIME=60D               FILE.TIME=60D
FILE.SIZE=100000000     FILE.SIZE=100000000         FILE.SIZE=100000000
FILE.MAX=1              FILE.MAX=1                  FILE.MAX=1
```

4.14.2 Average & Min/Max Compression Arrays

Only analog input (AI) and analog output (AO) registers can be archived using averaging and min/max compression. An AI is sampled and averaged over the LOG.RATE period. SCAN.RATE defines the AI sample time configuration table (see section *4.2 – Configuration Table*).

For the AI to be archived, the LOG.RATE must be an even multiple of the SCAN.RATE, and equal to or greater than the SCAN.RATE. For an AO register, the sample time is defined by whenever the XXXCODE control program, telemetry, or the keypad writes to the register. The AO interval average is integrated over the interval time to generate an accurate average for that interval. The avg/min/max values are guaranteed to process all AI and AO register values.

SAMPLES

In the following example, every five minutes the averaged value of AO5 and the min/max values are stored in archive array 1. The entries are retained for 60 days. The file size is to be no larger than 100M and stops if that value is reached because `FILE.MAX=1`.

```
BEGIN
ARRY.NUM=1
INPUT=AO5
ARRY.TYPE=COMPRESS
LOG.RATE=5M
ARRY.NAME="VFD1_SPEED"
FILE.TIME=60D
FILE.SIZE=100000000
FILE.MAX=1
```

Memory Requirements

1) The memory required to store the preceding sample array 1 is calculated as follows:

   Each avg/min/max entry uses 50 bytes, for a 150 byte total
   60 days = 5,184,000 seconds
   5 minutes = 300 seconds
   5,184,000 seconds / 300 seconds = 17,280 entries
   17,280 entries × 150 bytes/entry = 2.6M bytes

2) If the sample archive was set up with average values at a 5 minute rate and a min/max reading taken every hour, its configuration would be as follows:

```
BEGIN
ARRY.NUM=2
INPUT=AO5
ARRY.TYPE=COMPRESS
LOG.RATE=5M
ARRY.NAME=”VFD1_SPEED”
MN.MX.RATE=1H
FILE.TIME=60D
FILE.SIZE=100000000
FILE.MAX=1
```

   Because of the different rates of archiving the average and min/max values, its memory usage would be:

   Subtotal 1

   Average value entry requires 50 bytes
   (5,184,000 seconds / 300 seconds) = 17,280 entries for average
   17,280 entries × 50 bytes/entry = 864 Kbytes for average

   Subtotal 2

   Min/max value requires 100 bytes
   (5,184,000 seconds / 3,600 seconds) = 1,440 entries for min/max
   1,440 entries × 100 bytes/entry = 144 Kbytes for min/max

   Total Memory Required

   (864K + 144K) = 1,008 Kbytes

3) To archive only average values, the following configuration would be defined:

```
BEGIN
ARRY.NUM=3
INPUT=A05
ARRY.TYPE=COMPRESS
LOG.RATE=5M
ARRY.NAME=”VFD1_SPEED”
MN.MX.RATE=NONE
FILE.TIME=60D
FILE.SIZE=100000000
FILE.MAX=1
```

   The memory usage would be:

   Average value entry requires 50 bytes
   (5,184,000 seconds / 300 seconds) = 17,280 entries
   17,280 entries × 50 bytes/entry = 864 Kbytes

4.14.3 Change-Based Archive Arrays

Change based archiving is supported for analog input (AI), analog output (AO), digital input (DI), and digital output (DO) registers.

For DI or DO registers any state change causes an entry to be stored into the archive array. The DELTA.REG field does not apply when archiving DI or DO registers; they are archived with a timestamp.

An AI and AO register is archived if the absolute value of difference between the current and previous register values is equal to or greater than the delta value stored in the register as defined by DELTA.REG.

Each state change entry requires 5 bytes; a delta change entry requires 9 bytes.

CHANGE-BASED ARRAY SAMPLES

In the following example, a DI register is archived based on change:

```
BEGIN
ARRY.NUM=100
INPUT=DI3
ARRY.TYPE=CHNG.BASE
ARRY.NAME=”VALVE1_OPEN”
FILE.TIME=180D
FILE.SIZE=100000000
FILE.MAX=1
```

Each changed-based entry uses 50 bytes each; (100M/50 bytes) = 2M entries.

In this next example, an AI or AO register is archived on change. For an entry to be stored in the archive, the current value of AI3 (below) must be equal to or greater than 2.5 units from the last archived entry:

```
BEGIN
ARRY.NUM=100
INPUT=AI3
ARRY.TYPE=CHNG.BASE
ARRY.NAME=”VFD1_FDBK”
DELTA.REG=AO100
FILE.TIME=180D
FILE.SIZE=1000000
FILE.MAX=10
```

10M/50 bytes = 200K entries can be stored; this example results in 10 files of 1M each.

4.14.4 Alarm Summary Array

Alarm summary archiving automatically stores all alarm registers upon either a state change of the alarm register or the alarm sequence annunciator. The alarm register value, as well as the state of the alarm sequence annunciators, is recorded. In this way, all alarm conditions are easily monitored, logged and time stamped and an audit trail of all alarm activity can be generated and reported.

Each alarm register state change requires 8 bytes of memory.

SAMPLE—Below is a partial listing of a sample alarm summary archive array:

```
BEGIN
ARRY.NUM=100
INPUT=NONE
ARRY.TYPE=ALARM.SUM
ARRY.NAME="PS15_ALARMS"
FILE.TIME=60D
FILE.SIZE=1000000
FILE.MAX=10
```

Each alarm entry uses 50 bytes: 10M/50 bytes = 200K entries

4.14.5 Archive Data Retrieval

Archive arrays are stored on a USB flash drive in the following directory:

```
/fs/hd1-usbstick/-------_tables
```

The arrays are stored as .CSV files, with file names designated as:

```
<ARRY.NAME>_<ARRY.NUM>_<MDYHMS>.csv
```

Example:

```
PUMP_CALL_P2_100_031715172044.csv
```

There are two ways to retrieve archive arrays:

- Copy the data arrays from the USB flash drive to a PC for analysis.
- Log into the --------- using SFTP , then upload the arrays to the remote PC.

4.15. Modbus® and the Modbus® Slave Table

Communications can be over Ethernet using Modbus® TCP and/or serial communications using Modbus® RTU. The default register value is represented as a 16-bit unsigned integer. Thus all -------- floating point numbers are converted to an integer before being sent. A negative value is sent as 0, and a value greater than 65,535 is sent as 65,535. All floating point numbers are truncated.

In order to send/receive floating point values, a -------- Modbus® slave device must have a slave table. It assigns the Modbus® registers as floating point values and also maps -------- device registers to corresponding Modbus® registers.

Notes: If a register is not defined in the Modbus® slave table, then default integer rules apply.

Table 4-18 – Modbus Registers

--------	Modbus Device	Modbus Msg Type
Digital Outputs 0 – *n*	00001 – 01999	ROS,FSC
Digital Inputs 0 – *n*	10001 – 21999	RIS
Analog Inputs 0 – *n*	30001 – 31999	RIR
Analog Outputs 0 – *n*	40001 – 41999	ROR
Setpoints 0 – *n*	42001 – 43999	
Timer/Counters 0 – *n*	44001 – 45999	PSR
Index Registers	46001 – 47999	

The Modbus® and -------- register indices are offset by one; Modbus® starts with 1 (40001), whereas the -------- starts with 0 (AO0). Modbus® register 40000 is not a legal register index. In the --------, Modbus® 40001 register maps to AO0, while register 40002 maps to AO1.

4.15.1 -------- Modbus® Slave Table Entries

40010F = AO100 ; the -------- AO100 is sent and received as a float value

40011 is now an illegal register index because the float takes two integer registers

40012 = AO101 ; the -------- AO101 is integer because of omission of appended F

40013F = AO102 ; the -------- AO102 is sent and received as a float value

- Registers defined in the Modbus® slave table will not have any offset associated with the register pairing.
- Anything undefined in the Modbus® table remains offset by 1

00001 = DO0 ; -------- digital output 0 is mapped to Modbus® coil 1

00001 is WRITE ONLY and cannot be read

10001 = DI10 ; -------- digital input 10 is mapped to Modbus® status 1

10001 is READ ONLY and cannot be written

If a DI is set as a coil and also a status, the digital can be READ/WRITE

00020 = DI20 ; -------- digital input 20 is mapped to Modbus® coil 20

10020 = DI20 ; -------- digital input 20 is mapped to Modbus® status 20

The above combination lets DI20 be READ/WRITE

4.15.2 -------- Modbus® Master Entries

The -------- polling and quiescent tables define whether a register value is floating point by appending an F to the register index; it defaults as an integer. Please refer to the previous polling and quiescent table sections (4.6 and 4.5, respectively) for more information on Modbus® entries.

5.0 Programming

5.1 XXXCODE Programming

<company> XXXCODE is an English-like language that uses statements to implement a control strategy for the --------. Normally, the program is downloaded to the -------- via Ethernet or via the maintenance port (COM 1). Small programs can also be entered (or small program changes made) from the -------- keypad for units equipped with either the full-size OIT or color HDMI touchscreen.

5.1.1 Register Types

XXXCODE programs manipulate (read and write) several register types used to access physical inputs/outputs, hold intermediate calculations, time values, etc. With the exception of digital output registers, each can be either physical or internal. The number of each type is set in the configuration table. Register types are described as follows:

Analog Input

Each analog (level) input register can hold a floating point number in the range $\pm 3.37 \times 10^{38}$, using a 32-bit internal (IEEE standard) representation.

- **Physical**—This type has a corresponding physical analog input channel that can read a voltage or current level from a connected device. An uncalibrated physical analog input value is displayed as a whole number, corresponding to the raw data seen on the AI channel in 12-bit resolution (0 – 4095). When a XXXCODE program stores a value to such a register, it is overwritten when the analog input channel is scanned.
- **Internal**—This type has no corresponding input channel, and is typically used to hold a number for intermediate calculations or for other programming purposes.

Analog Output

Each can hold a floating point number in the range $\pm 3.37 \times 10^{38}$.

- **Physical**—This type has a corresponding physical analog output channel controlling a voltage or current level to drive a connected device. An uncalibrated physical analog output value is displayed as a whole number, corresponding to the raw data value being output on the AO channel in 12-bit resolution (0 – 4095).
- **Internal**—This type has no corresponding output channel, and can be used like an internal analog input for programming purposes.

Digital Input (Status)

- **Physical**—This type has a corresponding physical digital input channel that can sense an ON/OFF (closed/open contact) condition. When a XXXCODE program stores a value to such a register, the register reverts to its actual state when it is scanned. (The scan rate is set in the configuration table.)
- **Internal**—This type has no corresponding physical digital input channel, and can be used as a flag for programming purposes.

Digital Output

A digital output (pump) register is always associated with a physical output channel (there are no internal digital outputs). In response to program control, a digital output channel generates an ON/OFF (contact closed/open) condition typically used to control a pump, motor, light, or other device.

5.1.2 Timers/Counters

All timers/counters are in a continuous range of indexes, from T0 to T*n*. Each of the five timer/counter types corresponds with a subset of the total index range. As listed in *Table 5-1*, subsets are ordered by index, with pulse counters first and seconds timers last.

If no timers/counters of a given type are defined, then they don't consume any index range space. The distribution of timers/counters among the available types is set in the configuration table.

Table 5-1 – Timers/Counters

Type	Display	Internal	Max	Size	Range
Pulse Counter	1 count	1 count	16	32	4.2 billion
Hours Timer	0.001 hour (3.6 seconds)	2 seconds	256		272 years
HMS Timer	1 second	0.5 seconds			68 years
Event Counter	1 count	Program control or 0.02 seconds			4.2 billion
Seconds Timer	0.01 second	0.01 seconds			497 days

In general, all timers/counters can be:

- Preset
- Turned ON or OFF (enabled/disabled)
- Read
- Displayed

When a timer/counter reaches its maximum count value, it resets to 0 and continues running. Each type is defined as follows.

Pulse Counter

Pulse counters count pulses occurring on the first 16 digital inputs (DI). On a per-DI basis, a pulse counter increments once for each contact opening or closure and is counted at a rate of up to 1000 per second. Set in the configuration table, the *pulse map* value determines whether the rising or falling edge triggers an increment. The *pulse delay* setting determines the minimum pulse width that is seen.

Hours Timer

Keeping track of elapsed time, hours timers continuously increment when enabled and the -------- is in run mode. A blinking dot on the right-hand side of the display indicates that the timer is running. The current time value is held when disabled or the -------- is in standby mode. Hours timer values are stored internally as a 32-bit integral number, with each count representing 2 seconds. A HHHHH.FFF (hours and fractions thereof) format is used for display purposes and by a XXXCODE program.

HMS Timer

HMS timers continuously increment when enabled, regardless of whether the -------- is in run or standby mode. A blinking dot on the right-hand side of the display indicates that the timer is running. The current time value is held when disabled.

HMS timers are used in pairs: the first is used as a running timer, while the second holds a maximum value. When the first timer value reaches that of the second timer, the first resets to 0 and continues to run. Normally, the second counter is always off. When storing to, or retrieving from, an HMS timer using XXXCODE, values are represented as a 32-bit integral number of 0.5 second counts (example: 100 = 50 seconds). A HHHH.MM.SSS (hours, minutes, and seconds) display format is used.

Event Counter

XXXCODE statements can directly store to, read, and increment these general purpose counters. Event counters are stored as 32-bit, unsigned integer numbers having a maximum range of 0 – 4,294,967,295. When used to control a digital output (called a pulse DO), an event counter can automatically generate timed pulses. The -------- decrements the value stored in the associated event counter at a rate of 50 counts per second (50Hz), each count having a time delay of 20 milliseconds. The generated pulse width range is between 20 milliseconds and 994 days, with a resolution of 20 ms. Event counters 51 through 54 act as normal event counters and have no corresponding pulse DO.

A pulse DO is defined in the LED table by adding a trailing P to its line entry, as shown below:

```
LED81=DO21P
```

The first defined pulse DO is associated with the first event counter. Others are associated with an event counter which is at the same offset from the first pulse DO. The following illustrates this principle:

- The first pulse DO is defined as DO20
- The first event counter is 50 (set in the configuration table)
- If the next pulse DO is defined as DO25, then it is associated with event counter 55

Table 5-2 shows pulse DO/event counter operation under various starting conditions:

Table 5-2 – Pulse DO/Event Counter Operation

Pulse DO State	Counter State	Counter Value	Action	Result
OFF	OFF	0	Store non-zero value to counter	Counter decrements at 50 Hz; DO turns ON when count reaches 0
	ON			DO immediately turns ON; counter decrements at 50 Hz; DO turns on when count reaches 0
ON	OFF			DO immediately turns off; counter decrements at 50 Hz; DO turns ON when count reaches 0
OFF				Counter decrements at 50 Hz; DO turns OFF when count reaches 0

Note: The event counter ON/OFF state does not determine whether the counter is automatically decremented—it automatically decrements in either case. For an event counter used to time a pulse DO, the ON/OFF state determines whether it times the ON or the OFF duration. When switching an event counter having a 0 value from either state, the pulse DO output does not change state. For the pulse DO to also change state when changing the event counter state, the latter must have a non-zero value. This provides for a bumpless transfer of the pulse DO when timing both the ON and OFF times.

This XXXCODE example illustrates how to time both the ON and OFF states of a DO:

```
100- T55 ON AND T55 EQ 0           : Is it ready for OFF timing
ACTION T55 OFF AND SP100 STORE T55 : Turn pulse DO OFF for delay
110- T55 OFF AND T55 EQ 0          : Is it ready for ON timing
ACTION T55 ON AND SP101 STORE T55  : Turn pulse DO ON for delay
```

Seconds Timer

Seconds timers are similar to hours timers. When the -------- is in run mode, they continuously increment and keep track of elapsed time (in seconds and fractions thereof) when enabled. A blinking dot on the right-hand side of the display indicates that the timer is running. The current time value is held when disabled or the -------- is in standby mode. Seconds timer values are stored internally as a 32-bit integral number, with each count representing 0.o1 seconds. A SSSSSS.FF (seconds and fractions thereof) format is used for display purposes and by a XXXCODE program.

5.1.3 Setpoint

Setpoints are typically used to store operating parameters and constants, such as pump turn-on and turn-off levels. Stored in write-protected RAM for safety, each is an IEEE format 32-bit floating point number having a maximum range of $\pm 3.37 \times 10^{38}$. Setpoint values can be changed from the keypad using the CHANGE menu option.

5.1.4 Index

Index registers can be used for general purposes in XXXCODE programming. They also have a special function that lets them be used to index into other registers (setpoints, DIs, DOs, AIs, or AOs). For example, if index register 3 has a current value of 8, then the XXXCODE expression,

```
SETPOINT INDEX 3
```

accesses setpoint 8 (the index register value gets substituted). Index registers are stored as 32-bit unsigned integer numbers having a maximum range of 0 – 4,294,967,295.

5.2 XXXCODE Program and Statement Structure

XXXCODE has many constructs similar to BASIC. A XXXCODE program consists of one or more line entries, each having a unique line number identifier ranging from 0 – 65534. Each line number is immediately trailed by a hyphen, then one or more white spaces consisting of blank lines, spaces, or tabs). Unless a branching expression redirects flow, each program runs sequentially according to its line number. You can skip blocks of line numbers in order to reserve room for future code.

All code must be in upper case.

Comments are encouraged and consist of a colon (:) with trailing text. A comments can appear on a line by itself, or at the end of a code line.

The following (optional) line can be added to the end of each program:

```
END of program
   or
END
```

Each line entry consists of an optional conditional expression, evaluated as TRUE or FALSE, and a mandatory action expression. Only if the conditional expression is TRUE is the action expression then executed.

A sample XXXCODE line is shown below:

```
: Sample line entry with comment

10- LEVEL 0 < SETPOINT 7 ACTION PUMP 10 OFF  : Another comment
```

Register names are abbreviated as in the following examples (see *Table 5-3*):

```
10- L0 < SP7 ACTION P10 OFF : If Level 0 is less than Setpoint 7
                            :   then turn Pump 10 OFF
20- ACTION SP17 STORE T9    : STORE Setpoint 17 in Timer 9
```

Table 5-3 – Register Short Forms (Abbreviations)

Long Form	Short Form	Meaning
AO	AO	Analog Output
INDEX	IR	Index Register
LEVEL	L	Analog Input
PUMP	P	Digital Output
SETPOINT	SP	Setpoint
STATUS	S	Digital Input
T	T	Timer/Counter

5.3 Mathematical Functions

XXXCODE supports many mathematical functions, as shown in *Table 5-4*. Parenthetical subexpressions are evaluated first, otherwise all expressions are evaluated from left to right.

Table 5-4 – XXXCODE Mathematical Functions

Name/ Symbol	Function	Example
+	addition	SP3 + L1
-	subtraction	L10 – L8
x	multiplication	AO7 X SP1
/	division	L9/SP77
EXP	inverse natural log (e)	EXP L3
LN	ln natural log	LN AO6
SIN	sine	SIN L88
COS	cosine	COS AO99
TAN	tangent	TAN L13
SQRT	square root	ACTION SQRT AO 19 STORE AO 20
AND	logical and	L4 < SP7 AND P18 ON
AND	statement concatenation	ACTION P7 ON AND P8 OFF
OR	logical or	L3 < SP5 OR L3 > SP6 ACTION P5 OFF
NOT	logical not	NOT (L8 > SP5) ACTION P5 ON
XOR	logical exclusive	OR L5 < SP8 XOR L6 < SP9 ACTION P1 ON
MIN	minimum	ACTION L1 MIN L2 STORE L3
MAX	maximum	ACTION AO9 MAX SP5 STORE AO9
≥	greater than or equal to	L1 ≥ 3.0 ACTION P3 ON
≤	less than or equal to	L1 ≤ 3.0 ACTION P3 ON
<	less than	L1 < 5.5 ACTION P3 OFF
>	greater than	L1 > 3.0 ACTION P3 ON
EQ	equality	T17 EQ T18 ACTION P77 ON
NEQ	not equal	T20 NEQ T21 ACTION P 15 ON
( )	left/right parenthesis	SP17 / (T45 + T34)

Conditional Expressions

Conditional TRUE/FALSE expressions are optional and determine if a trailing action expression (not shown) is to be executed. Examples:

```
100- L29 < SP9              : If Level 29 is less than Setpoint 9
110- T17 ON AND P18 OFF     : If Timer 17 is ON and Pump 18 is OFF
```

Action Expressions

An action expression performs some task; it is optionally preceded by a conditional expression. Examples:

```
200- ACTION P1 ON           : Turn Pump 1 ON
210- ACTION SP23 STORE T55 AND SP24 STORE T56
                            : Store Setpoint 23 to Timer 55
                            :   and Store Setpoint 24 to Timer 56
220- L29 < SP9 ACTION P1 ON
                            : If Level 29 is less than Setpoint 9
                            :   then turn Pump 1 ON
```

Storing Values

Use the STORE command to cause a value, or an expression result, to be stored in to a register. Examples:

```
230- ACTION L12 + SP17 STORE T19     : Store Level 12 plus
                                     :   Setpoint 17 to Timer 19
300- ACTION SP88 STORE AO7           : Store Setpoint 88
                                     :   to Analog Output 7
420- ACTION COS AO77 + SP24 STORE AO78
                                     : Store the cosine of Analog
                                     :   Output 77 plus Setpoint 24
                                     :   to Analog Output 78
```

Control Flow

Three commands let you branch program execution flow:

```
GOTO
GOSUB
RETURN
```

GOTO is used in the following example:

```
30- ACTION GOTO 99      : Direct program flow to line number 99
```

This line causes execution to jump to line 99. If it does not exist, then execution continues with the next line. If there is none, then execution moves to the program beginning.

GOSUB is used in a similar way:

```
30- ACTION GOSUB 99      : Go to the subroutine that begins on line 99,
                         :   then return execution to the line following
                         :   line 30
```

GOSUB remembers where it originated and returns to the next XXXCODE line when it encounters a RETURN. This is useful for writing one or more subroutines that can be accessed from multiple locations in a single XXXCODE program, each time returning to its source when the subroutine has been executed.

Example:

In the following example, line 10 calls the subroutine located on line 99. When the RETURN is encountered, execution resumes at line 20. The identical subroutine is called again at line 50, returning execution to line 60 once it has completed execution (for a second time in this example).

```
10- ACTION GOSUB 99                    : Jump to subroutine at line 99
20- L17 < SP67 ACTION SP34 STORE T88
30-      …
40-      …
50- ACTION GOSUB 99                    : Jump to subroutine at line 99
60-      …
70-      …
80- ACTION GOTO 10                        : Start all over again (loop)
                                          : Program ends at line 80
                                          :
99- ACTION P17 ON AND P13 OFF AND RETURN  : A repeatable subroutine
```

5.4 XXXCODE Execution Trace

To assess functionality and help track down problems, a method exists to trace program execution. Providing detail, executed line information can be sent to the standard output device (e.g., a printer or maintenance port as defined in the configuration table). Usually, sending it to the maintenance port is a good choice, as it can be viewed on an attached terminal (or a PC emulating a terminal).

To enable this function in the configuration table (see section 4.2.3.9), set the TRACE Enable entry within the MONITOR category to YES. The S/S LED flashes to alert the operator whenever TRACE is enabled.

Four SPEC.REGS entries (see section 4.2.3.6) control its operation:

DEBUG.SP—A line number can be put into this setpoint (default SETPOINT 125) to cause the SBY LED to flash when that line is executed. In addition, if TRACE1.DI is ON, all information about the line number in the setpoint is reported.

TRACE1.DI—Holds the DI register index used to enable/disable the tracing of XXXCODE line number information. When this DI is ON, the following information is sent to the output device:

- *nnnn* executed XXXCODE line number
- A ACTION portion of the executed expression
- E ELSE portion of the executed expression

TRACE2.DI—Holds the DI register index used to enable/disable additional program line tracing information (beyond the TRACE1 level).

TRACE.IR—Used to slow down a program so as to monitor it using one of the trace functions, the trace delay index register equates to the number of milliseconds allowed per XXXCODE line. For example, if 50 is stored and a XXXCODE line takes 30 ms to execute, then a 20 ms delay is imposed before ceding control to the next line. Default: IR 73.

5.5 Examining Program

Using the keypad, a XXXCODE program can be examined by doing the following:

1. Select the SPECIAL ► EXAM PGM ► MAIN PGM menu option.
2. If there is a XXXCODE program, the first line number is displayed e.g., LN10)
3. To examine the line, press [ENTER]. It scrolls across the display.
4. To move to another line number, use the ▲ ▼ keys.
5. To conclude viewing the program, press [EXIT].

5.6 Change the Program from the Keypad

A XXXCODE program can be changed only from the full 32-key keypad or the color HDMI touchscreen interface. A program can be downloaded using ---------- 64 (*Chapter 8—----------h 64 IDE*).

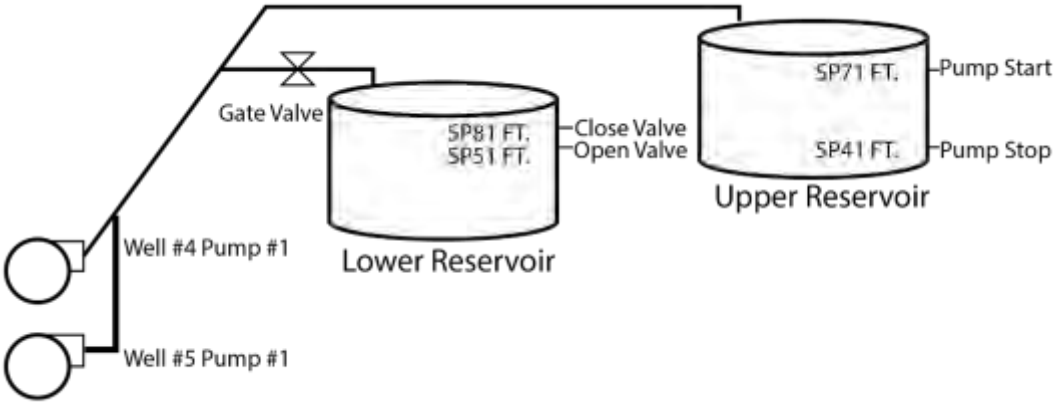
To change a XXXCODE program:

1. Enter program mode (10.4 [ENTER]).
2. Select MAIN PGM. If there is an existing program, the first line number is shown. Other lines can be viewed in the same manner as described in section 5.5 (above).
3. To insert a new XXXCODE line, press [STORE] <*line number*> [ENTER] or, to replace the current line, press [STORE] [ENTER].
4. (If inserting a new line) Key in the desired program statement, followed by [ENTER].
5. To leave program mode, press [EXIT] several times.

5.7 Sample XXXCODE Program

The following is an aid in learning how to both write and interpret a XXXCODE program. It is derived from a real application and is presented along with descriptive material.

Figure 5-1 – Sample XXXCODE Program Scenario



In this scenario, the -------- does the following:

- Starts and stops well pumps
- Opens and closes the gate valve based on the lower and upper reservoir levels
- Monitors reservoir levels, generating high and low alarms
- Automatically purges based on auto-purge setpoints

Automatic Well Pump Control

The -------- senses the lower and upper reservoir levels at analog inputs 0 and 1, respectively. The upper reservoir bottom sits at a higher elevation than that of the lower reservoir. This causes the lower reservoir to fill to the top, ahead of the upper reservoir—as illustrated above. The following control scheme is implemented to maintain both reservoir levels:

1. When the level drops below the pump start setpoint (SP41 ft.) value, the pump(s) in automatic mode start and the reservoirs begin to simultaneously fill. The gate valve should be open if the open valve setpoint (SP51 ft.) is set to an elevation above that of the pump start setpoint.
2. The lower reservoir level reaches the close valve setpoint (SP81 ft.) and the gate valve closes, diverting all pump flow to the upper reservoir.
3. The upper reservoir level reaches the pump stop setpoint (SP71 ft.) and the pump(s) stops.
4. When the lower reservoir level again drops below the open valve setpoint (SP51 ft.), the gate valve reopens.
5. When the upper reservoir level again drops below the pump start setpoint (SP41 ft.), the cycle starts over.

Alarm Reset

This sample program uses two alarm types: Manual Reset and Auto Reset. When the -------- generates either, both the common and new alarm LEDs flash. The common alarm always indicates that a new alarm has been generated and has not yet been acknowledged or reset. The following describes what happens next, depending on the generated alarm type and which reset type is used—either the [Manual Reset] button (mounted externally) or the ALRM ACK menu option.

Manual Reset

The sample program uses two Manual Reset alarm types:

- Level Transducer Fail (lower or upper Reservoir, low or high)
- Pump Fail (well #4, pump #1 or well #5, pump #1)

ALRM ACK menu option:

1. The common LED is extinguished.
2. The new alarm LED changes from flashing to a steady illuminated state.

[Alarm Reset] is pressed:

1. The common LED is extinguished.
2. If the alarm condition still exists, then the LED changes to a steady (non-flashing) state. Once the condition has cleared, [Alarm Reset] must be pressed. If the condition no longer exists, the alarm is cleared and the LED is extinguished.

Auto Reset

The sample program uses two Auto Reset alarm types:

- High-Level Alarm
- Low-Level Alarm

ALRM ACK menu option or [Alarm Reset] pressed:

1. The common LED is extinguished.
2. If the alarm condition still exists, then the LED changes to a steady (non-flashing) state. Once the condition no longer exists, the alarm is cleared and the LED is extinguished. If the condition clears before the alarm is acknowledged or reset, the LED continues to flash until a reset type is deployed, after which it is extinguished.

Table 5-5 – Sample Program Parameters

Alarm Name	Generating Condition	Delay Setpoint	Internal Alarm Status	Alarm Type
Upr Reservoir Transducer Fail High	AI 1 > SP99 ft.	SP600 sec	S300	MANUAL
Upr Reservoir High Level	AI 1 > SP98 ft.	SP601 sec	S301	AUTO
Upr Reservoir Low Level	AI 1 < SP97 ft.	SP602 sec	S302	
Upr Reservoir Transducer Fail Low	AI 1 < SP96 ft.	SP603 sec	S303	MANUAL
Lwr Reservoir Transducer Fail High	AI 0 > SP95 ft.	SP604 sec	S304	
Lwr Reservoir High Level	AI 0 > SP94 ft.	SP605 sec	S305	AUTO
Lwr Reservoir Low Level	AI 0 < SP93 ft.	SP606 sec	S306	
Lwr Reservoir Transducer Fail Low	AI 0 < SP92 ft.	SP607 sec	S307	MANUAL
Well #4 Pump #1 Fail	Pump called and no running status received	SP608 sec	S308	
Well #5 Pump #1 Fail		SP609 sec	S309	
Common Transducer Fail Status	Any transducer fail high or low status on	No Delay	S311	

Effect of Alarms on Pump and Valve Control

Some of the alarms in Table 5-1 stop and lock out the well pumps, or close and lock out the gate valve. These are listed below:

Lower Reservoir Level Transducer Fail—If a lower reservoir transducer fail high/low alarm condition exists, then the lower reservoir level is not known. The gate valve is closed and remains in that state until the condition clears and the alarm is reset. Upper reservoir well pump and level control is not affected since the gate valve is closed and the lower reservoir cannot be filled any higher.

Upper Reservoir Level Transducer fail—If an upper reservoir transducer fail high/low alarm condition exists, then the upper reservoir level is not known. The well pumps are stopped and remain in that state (unless manually started) until the condition clears and the alarm is reset.

Well #4/5 Pump #1 Fail—The corresponding pump is locked out until the alarm is cleared.

Purge of *<company>* Reactive Air Level Monitoring System

The system is capable of initiating both scheduled purges and manual purges (defined below).

Auto Purge—The -------- initiates a purge cycle every SP37 hours. Once initiated, the cycle executes as follows:

1. The lower reservoir level storage register (AO89) is held at its current value.
2. The purge solenoid energizes and then starts after 2 seconds.
3. The purge compressor runs for SP181 seconds and then stops.
4. The purge solenoid remains energized for SP182 seconds afterward.
5. After a settle delay of SP183 seconds, the -------- begins updating the lower reservoir level storage register (AO89), used for gate valve control.

Manual Purge—The [Manual Purge] button is mounted external to the --------. When pressed, a purge cycle is initiated and the purge solenoid is energized. The button must be released before the cycle advances to the next step (step 3, above). Once the purge compressor starts, you can extend the cycle by pressing and holding [Manual Purge]. The solenoid and compressor remain energized and running until you release the button, at which time the cycle resumes at step 3 of the sequence.

Note: When using the ---------- Reactive Air System, typical purge time values are 4 to 6 seconds, and the purge interval is 16 to 36 hours. Significantly decreasing the interval or increasing the time might cause excessive wear to the reactive air system.

Sample XXXCODE Program Code

```
:------ ------- WATER DISTRICT
:GROUND WATER STORAGE RESERVOIR PROJECT
:
:— INITIALIZE ————————————————————————
00- ACTION T0 ON AND T400 INCR         :------ RUNTIME & STARTS

:— ALARM RESET ———————————————————————
02- S23 ON                    :RESET BUTTON PRESSED, CLEAR
ACTION S300 OFF               :..UPPER LEVEL XD FAIL HIGH
AND S303 OFF                  :..UPPER LEVEL XD FAIL LOW
AND S304 OFF                  :..LOWER LEVEL XD FAIL HIGH
AND S307 OFF                  :..LOWER LEVEL XD FAIL LOW
AND S308 OFF                  :..WELL #4 PUMP #1 FAIL
AND S309 OFF                  :..WELL #5 PUMP #1 FAIL

:— TIMERS ————————————————————————————
10- ACTION T600 ON AND T601 ON AND T602 ON   :ALARM TIMERS ALWAYS ON
   AND T603 ON AND T604 ON AND T605 ON       :..
   AND T606 ON AND T607 ON AND T608 ON       :..
   AND T609 ON AND T610 ON                   :..
   AND T631 ON AND T632 ON                   :..& BACKSPIN TIMERS
   AND T37 ON                                :..& AUTO PURGE TIMER

11- S1 ON ACTION 0.0 STORE T631              :WELL #4 PUMP #1 BACKSPIN
12- S2 ON ACTION 0.0 STORE T632              :WELL #5 PUMP #1 BACKSPIN

13- T639 > SP639                             :SEQUENTIAL START DELAY
   ACTION T639 OFF AND 0.0 STORE T639        :..

:— METER PEGS ————————————————————————
50- L0 > 0.0 ACTION L0 STORE AO30            :LEVEL NOT NEG, STORE IT
   ELSE ACTION 0.0 STORE AO30                :..ELSE STORE 0.0
51- L1 > 0.0 ACTION L1 STORE AO31            :LEVEL NOT NEG, STORE IT
   ELSE ACTION 0.0 STORE AO31                :..ELSE STORE 0.0

:— PURGE ——————————————————————————————
90- T37 > SP37                               :IF PURGE TIME
   OR S22 ON                                 :..OR MANUAL PURGE
   AND P6 OFF                                :..& SOLENOID NOT CALLED
   ACTION 0.0 STORE T37                      :..RESET PURGE INTERVAL
   AND P6 ON                                 :..& PURGE S.V.
91- S22 ON ACTION 0.0 STORE T37              :DONT CONTINUE UNTIL RELEASE
                                             :..& EXTEND IF COMP. ON
92- T37 > ( 2.0 / 3600.0 )                   :AFTER 2 SECONDS
   AND T37 < ( SP181 / 3600.0 )              :..& INTERVAL NOT DONE
   AND P6 ON                                 :..& SOLENOID CALLED
   ACTION P7 ON                              :..& CLOSE COMPRESSOR ON
93- ( 2.0 + SP181 ) / 3600.0 < T37           :IF TIME
   ACTION P7 OFF                             :..COMPRESSOR OFF
94- SP181 + SP182 / 3600.0 < T37             :IF TIME
   ACTION P6 OFF                             :..SOLENOID VALVE OPEN
95- SP181 + SP182 + SP183 / 3600.0 < T37     :IF TIME
   ACTION L0 STORE AO89                      :..TAKE SAMPLE
```

```
:— ALARM GENERATION ————————————————————————
100- L1 < SP99                          :UPPER LEVEL IS VALID
   ACTION 0.0 STORE T600                :..RESET ALARM DELAY
   ELSE                                 :..ELSE,
   T600 > SP600                         :..IF TIME
   ACTION S300 ON                       :UPPER LEVEL XD FAIL HIGH
   101- L1 < SP98                       :UPPER LEVEL IS NOT HIGH
   ACTION 0.0 STORE T601                :..RESET ALARM DELAY
   AND S301 OFF                         :.. & ALARM STATUS
   ELSE                                 :..ELSE,
   T601 > SP601                         :..IF TIME
   ACTION S301 ON                       :UPPER LEVEL HIGH ALARM
102- L1 > SP97                          :UPPER LEVEL IS NOT LOW
   ACTION 0.0 STORE T602                :..RESET ALARM DELAY
   AND S302 OFF                         :.. & ALARM STATUS
   ELSE                                 :..ELSE,
   T602 > SP602                         :..IF TIME
   ACTION S302 ON                       :UPPER LEVEL LOW ALARM
103- L1 > SP96                          :UPPER LEVEL IS VALID
   ACTION 0.0 STORE T603                :..RESET ALARM DELAY
   ELSE                                 :..ELSE,
   T603 > SP603                         :..IF TIME
   ACTION S303 ON                       :UPPER LEVEL XD FAIL HIGH
104- AO89 < SP95                        :LOWER LEVEL IS VALID
   ACTION 0.0 STORE T604                :..RESET ALARM DELAY
   ELSE                                 :..ELSE,
   T604 > SP604                         :..IF TIME
   ACTION S304 ON                       :LOWER LEVEL XD FAIL HIGH
105- AO89 < SP94                        :LOWER LEVEL IS NOT HIGH
   ACTION 0.0 STORE T605                :..RESET ALARM DELAY
   AND S305 OFF                         :.. & ALARM STATUS
   ELSE                                 :..ELSE,
T605 > SP605                            :..IF TIME
   ACTION S305 ON                       :LOWER LEVEL HIGH ALARM
106- AO89 > SP93                        :LOWER LEVEL IS NOT LOW
   ACTION 0.0 STORE T606                :..RESET ALARM DELAY
   AND S306 OFF                         :.. & ALARM STATUS
   ELSE                                 :..ELSE,
   T606 > SP606                         :..IF TIME
   ACTION S306 ON                       :LOWER LEVEL LOW ALARM
107- AO89 > SP92                        :LOWER LEVEL IS VALID
   ACTION 0.0 STORE T607                :..RESET ALARM DELAY
   ELSE                                 :..ELSE,
   T607 > SP607                         :..IF TIME
   ACTION S307 ON                       :LOWER LEVEL XD FAIL HIGH
108- P1 OFF OR S1 ON                    :NOT CALLED OR IS RUNNING
   ACTION 0.0 STORE T608                :..RESET ALARM DELAY
   ELSE                                 :..ELSE,
   T608 > SP608                         :..IF TIME
   ACTION S308 ON                       :WELL #4 PUMP #1 FAIL
109- P2 OFF OR S2 ON                    :NOT CALLED OR IS RUNNING
   ACTION 0.0 STORE T609                :..RESET ALARM DELAY
   ELSE                                 :..ELSE,
   T609 > SP609                         :..IF TIME
   ACTION S309 ON                       :WELL #5 PUMP #1 FAIL
```

```
111- S300 ON OR S303 ON                :ANY TRANSDUCER FAIL STATUS
   OR S304 ON OR S307 ON               :..
   ACTION S311 ON                      :COMMON XD FAIL STATUS ON
   ELSE ACTION S311 OFF                :..ELSE, OFF

:— PUMP CALLS BASED ON UPPER RESERVOIR LEVEL ————————————
200- L1 > SP71 ACTION S31 OFF
   AND S32 OFF
201- L1 < SP41 ACTION S31 ON
   AND S32 ON

:— VALVE OPEN / CLOSE FLAG BASED ON LOWER RESERVOIR LEVEL ———
   210- AO89 > SP81 ACTION S51 OFF
   211- AO89 < SP51 ACTION S51 ON

:— COMMON PUMP STOP FLAG ————————————————————
290- S300 ON                           :UPPER LEVEL XD FAIL ALARM
   OR S303 ON                          :..STATUS (HI OR LO)
   ACTION S60 ON                       :COMMON STOP FLAG ON
   ELSE ACTION S60 OFF                 :..ELSE, OFF

:— COMMON VALVE CLOSE FLAG ————————————————————
291- S304 ON :LOWER LEVEL XD FAIL ALARM
   OR S307 ON                          :..STATUS (HI OR LO)
   ACTION S61 ON                       :COMMON CLOSE FLAG ON
   ELSE ACTION S61 OFF                 :..ELSE, OFF

:— WELL #4 PUMP #1 START / STOP ——————————————————
300- S31 ON                            :PUMP IS REQUIRED
AND S1 OFF :..& NOT RUNNING
   AND T631 > SP631                    :..& BACKSPIN IS OK
   AND T639 OFF                        :..& SEQUENTIAL START IS OK
   AND S308 OFF                        :..& NO PUMP FAIL ALARM
   AND S60 OFF                         :..& NO COMMON STOP FLAG
   AND P1 OFF                          :..& PUMP NOT CALLED
   ACTION P1 ON                        :CALL THE PUMP
   AND T639 ON                         :..& SEQUENTIAL START DELAY
301- S31 OFF                           :PUMP IS NOT REQUIRED
   OR S308 ON                          :..OR IS FAILED
   OR S60 ON                           :..OR COMMON STOP FLAG ON
   ACTION P1 OFF                       :STOP THE PUMP

:— WELL #4 PUMP #1 START / STOP ——————————————————
302- S32 ON                            :PUMP IS REQUIRED
   AND S2 OFF                          :..& NOT RUNNING
   AND T632 > SP632                    :..& BACKSPIN IS OK
   AND T639 OFF                        :..& SEQUENTIAL START IS OK
   AND S309 OFF                        :..& NO PUMP FAIL ALARM
   AND S60 OFF                         :..& NO COMMON STOP FLAG
   AND P2 OFF                          :..& PUMP NOT CALLED
   ACTION P2 ON                        :CALL THE PUMP
   AND T639 ON                         :..& SEQUENTIAL START DELAY
303- S32 OFF                           :PUMP IS NOT REQUIRED
   OR S309 ON                          :..OR IS FAILED
   OR S60 ON                           :..OR COMMON STOP FLAG ON
   ACTION P2 OFF                       :STOP THE PUMP
```

```
:— OPEN / CLOSE VALVE COMMAND ————————————————
310- S51 ON                              :OPEN FLAG IS ON
   AND S61 OFF                           :..& NO COMMON CLOSE FLAG
   ACTION P8 ON AND P9 OFF               :OPEN THE VALVE
311- S51 OFF                             :OPEN FLAG IS OFF
   OR S61 ON                             :..OR COMMON CLOSE FLAG
   ACTION P8 OFF AND P9 ON               :CLOSE THE VALVE

:— PUMP START COUNTERS & RUNTIMERS ———————————————
:WELL #4 PUMP #1
400- S1 ON AND T1 OFF                    :PUMP RUNNING & TIMER OFF
   ACTION T1 ON AND T401 INCR            :..TIMER ON & INCR STARTS
401- S1 OFF ACTION T1 OFF                :PUMP NOT RUNNING, TIMER OFF
   :WELL #5 PUMP #1
402- S2 ON AND T2 OFF                    :PUMP RUNNING & TIMER OFF
   ACTION T2 ON AND T402 INCR            :..TIMER ON & INCR STARTS
403- S2 OFF ACTION T2 OFF                :PUMP NOT RUNNING, TIMER OFF
499- ACTION GOTO 02                      :RETURN TO START
END OF PROGRAM
```

5.8 Program Macro Keys

A XXXCODE program can be associated with a macro key, executing when that key is pressed.

Line numbers 1 – 8 are reserved for single line statements corresponding to each macro key. Each macro key accepts multiple XXXCODE lines and is configured as follows:

```
1 - EXAM LEVEL 1 ENTER
```

Each time macro key [1] is pressed, the value in analog input LEVEL 1 is displayed. The line numbers used in the macro program correspond to the macro key numbers.

Table 5-6 – Macro Keys

Macro Key	Line Numbers
1	100 – 199
2	200 – 299
3	300 – 399
4	400 – 499
5	500 – 599
6	600 – 699
7	700 – 799
8	800 – 899

XXXCODE lines numbered 1000 and above are used for macro key subroutines, as in:

`ACTION GOSUB 1000` and `ACTION RETURN`

Two new XXXCODE tokens have been added to the language:

KYBD ; the RTU waits for keypad input,
; the keypad LOCK timeout terminates the macro if waiting on keypad input

STRING ; up to 30 characters—the string must be in double quotes

The tokens are used as follows:

```
100- ACTION EXAM "ENTER FLOW GPM" AND KYBD STORE AO10
110- ACTION EXAM "DONE"
```

Line 100 scrolls ENTER FLOW GPM on the display for a short duration before it displays the AO10 value. The RTU now waits for keypad input to modify the value.

- Press the macro key a second time to repeat this display cycle.
- Press [EXIT] to cancel the macro.
- Press [ENTER] to store the AO10 value; DONE is briefly displayed (per line 110).

5.8.1 Examine the Current Macro Program

You can examine the current macro program from the keypad by doing the following:

1. Select SPECIAL ► EXAM PGM ► MACRO.PGM.
2. Press the ▲ ▼ keys to select the desired macro.
3. Press [ENTER] to scroll see the current program across the display.
4. Press [EXIT] to conclude macro viewing.

5.8.2 Change the Macro Program from the Keypad

Only the full keypad lets you program macro keys. They may also be downloaded using ------- 64 (see *Chapter 8 – ---------- 64 IDE*). To change macro programming from the keypad:

1. Enter program mode (10.4 [ENTER]).
2. Select MACRO.PGM.
3. Press [STORE] <*macro number*> [ENTER].
4. Key in the desired program statement, followed by [ENTER].
5. Press [EXIT] several times to leave program mode.

5.8.3 Macro Program in a File

Macro keys can be programmed from a file using ---------- 64, in a manner similar to how a main XXXCODE program is created. Macro program syntax is the same, with the exception that EXAM, KYBD, and display is permitted. This lets a register value or string be viewed on the display, with KYBD retrieving operator input. Line numbers in the file correspond to the respective macro key. This example shows a macro program for macro keys 3 and 4:

```
3- S19 ON AND S23 OFF AND T17 ON
      ACTION P23 OFF AND T17 OFF AND EXAM P23
      ELSE ACTION P23 ON AND P19 ON AND EXAM P23
4- ACTION 32.0 STORE AO54 AND
      33.0 STORE AO55 AND 34.0 STORE AO56 AND 35.0 STORE AO57
```