Introduction to Machine Edition Programming

Objectives

Upon completion of this module, you should be able to:

- Understand Micro 64 Software Architecture
- Create a Simple Ladder Diagram Program
- Download a Program to the Micro 64 Controller
- Monitor a Program in the Micro 64 Controller
Overview

Logic Developer PLC has a complete set of tools that can be used to develop control applications for the complete line of GE Fanuc controllers. The basic steps in developing a controller logic application are as follows:

- Create Logic Blocks that make up the Logic Application
- Create Block Logic in the desired application language such as Ladder Diagram or Instruction List
- Validate the Logic Application for any Warnings and Errors
- Download the Logic Application to the Controller
- Monitor the Logic Application in the Controller
- Perform Online modifications of the Logic Application in the Controller until it operates as desired

In order to develop applications for GE Fanuc controllers, it is first important to understand the basic software architecture for GE Fanuc controllers.

Micro 64 Application Program Structure

Each Micro 64 application is made up of a set of logic blocks that are managed in the Project tab of the Navigator. You use these logic blocks to organize your application. The application consists of a main block, named _MAIN, which is executed once each controller scans. This _MAIN block may call other blocks to execute logic functions as you desire.

The basic Micro 64 application program structure is illustrated in Figure 4-1.
The _MAIN block is created automatically for you when you create a Micro 64 target. The Micro 64 block-structured program always includes a _MAIN block. Program execution begins with the _MAIN block. Counting the _MAIN block, the program can contain up to 64 blocks.

<table>
<thead>
<tr>
<th>Block Type</th>
<th>Local Data</th>
<th>Programming Languages</th>
<th>Size Limit</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Block</td>
<td>Has its own local data</td>
<td>Ladder Diagram, Instruction List</td>
<td>16 KB</td>
<td>0 inputs 1 output</td>
</tr>
</tbody>
</table>

Table 3 - 1. Micro 64 Block Types

TIP!
The block type is displayed and may be modified in the Block Properties displayed in the Inspector.

All Micro 64 block types automatically provide an OK output parameter. The name used to reference the OK parameter within a block is Y0. Logic within the block can read and write the Y0 parameter. When a block is called, its Y0 parameter is automatically initialized to TRUE. This will result in a positive power flow out of the block call instruction when the block completes execution, unless Y0 is set to FALSE within the logic of the block.

TIP!
The block name may consist of 1 – 32 characters.

Program Blocks
Any block can be a Program Block. The _MAIN program block is automatically declared when you create a Micro 64 target. When you declare any other program block, you must assign it a unique block name. A program block is automatically configured with no input parameters and one output parameter (OK).

When a block-structured program is executed, the MAIN block is automatically executed. Other program blocks execute when called from the program logic in the MAIN block or another block. In the following example, if %I00001 (Power_UP) is ON, the program block named set-up will be executed:

![Figure 3 - 2. Example of Program Block Call](image-url)
Creating Logic Blocks

You create logic Blocks using the Project Navigator. To create a new Block, select the Logic Program Blocks node, and then select the New > LD Block menu. Then select the type of block that you want to create, an LD block for example.

This will create a new Ladder Diagram block with a default name LDBK. You can rename this block to give it a description of the logic that it contains.

Blocks may be programmed in a number of different application languages: Ladder Diagram (LD) or Instruction List (IL).

Ladder Diagram (LD) Logic

Logic written in Ladder Diagram language consists of a sequence of rungs that execute from top to bottom. The logic execution is thought of as “power flow”, which proceeds down along the left “rail” of the ladder, and from left to right along each rung in sequence.

The flow of logical power through each rung is controlled by a set of simple program instructions that work like mechanical relays and output coils. Whether or not a relay passes logical power flow along the rung depends on the content of a memory location (Variable) assigned to the relay. A relay will pass positive power flow if its associated memory location contains the value 1 (ON). The same relay passes negative power flow if the memory location contains the value 0 (OFF).
Usually an instruction that receives negative power flow does not execute and propagates the negative power flow on to the next instruction in the rung. However, some instructions such as timers and counters execute even when they receive negative power flow, and may even pass positive power flow out. Once a rung completes execution, with either positive or negative power flow, power flows down along the left rail to the next rung.

Within a rung, there are many functions that are part of the standard GE Fanuc function library that may be executed. These instructions can be used for operations like moving data stored in memory, performing math operations, and controlling communications between the CPU and other devices in the system. Some program functions, such as the Jump function and Master Control Relay, can be used to control the execution of the program itself. Together, this large group of Ladder Diagram instructions and standard GE Fanuc library functions makes up the instruction set of the CPU.

Creating Ladder Diagram (LD) Logic

You create Ladder Diagram logic in the grid-based LD Editor. You can place ladder diagram elements in the grid using the keyboard, Toolbar, and Toolchest. Once you place a logic element in the editor, you can then assign an application variable to the element. Variables can be assigned directly in the editor as well as by dragging-and-dropping them from the variable list.

The LD editor has a set of configuration options to define the editor’s display, such as Font, Colors, View Zoom, and Variable Display Options.

TIP!

The LD view options may be modified using the Editor’s View menu.

Inserting LD Instructions

There are several methods for creating Ladder Diagram logic in the grid-based LD Editor using the Right-click menu, Toolbar, Toolchest, and the Keyboard.

Right-Click (Basic) Method

To use the Right-click menu method, right-click the empty cell that will be the top-left cell occupied by the new instruction and choose Place Instruction. A smart list will now appear that contains all of the valid instructions. Now Type or choose from the list the mnemonic you want and press ENTER. The instruction now appears in the LD logic. If necessary, configure the instruction’s parameters.
Toolbar Method
To use the Toolbar method, Display the **Ladder Instructions** toolbar, if it is not already displayed. Click a tool button on the Ladder Instructions toolbar. Now in the LD editor, click a cell that will be the top-left cell of the new instruction. The instruction corresponding to the selected toolbar button appears in the LD logic. If necessary, configure the instruction’s parameters.

Repeat as required and then click the Pointer tool button to return to conventional editing.

Toolchest Method of Inserting LD Instructions
To use the Toolchest method, open the **PLC LD Instructions** drawer. Each folder in the drawer contains several closely related instructions. Expand the folder containing the instruction you want. Click an instruction mnemonic and drag it to the LD editor. Drop the instruction on the cell that will be the top-left cell for the new instruction. If necessary, configure the instruction’s parameters.
Keyboard Method
To use the Keyboard method, click the empty cell that will be the top-left cell occupied by the new instruction(s) and then begin typing the mnemonic of the first instruction. A smart list will appear that will contain the list of valid instructions. Type the mnemonic(s) and operand(s) you want, and then press ENTER.

Typing One Instruction
To enter a single instruction, type a mnemonic followed by operands with spaces separating each item. Put the operands in the following order: input 1, input 2, input n, output 1, output 2, output n. For example, typing "ADD_DINT MyDINT MyDINT2 Output " yields the following LD logic:

![Figure 3 - 5. Keyboard Method Example - Single Instruction](image)

Typing an Instruction With an Address and/or Length
If an instruction requires an address (the ??? operand) and/or a length (the ?? operand), type the instruction in the following order: mnemonic, address, length, input 1, input 2, input n, output 1, output 2, output n. For example, " MOVE_INT 1 MyReal MyAQ " inserts a MOVE_INT instruction with length 1, MyReal as the input, and MyAQ as the output:
Using shorthand for mnemonics

You may also use shorthand to specify instructions. The smart list interprets shorthand strings by substituting mnemonics that are the closest alphabetical match. For example, typing "AD MyDINT MyDINT2 Output" yields the same LD logic as typing "ADD_DINT MyDINT MyDINT2 Output".
Entering multiple instructions
To enter multiple instructions, simply separate each instruction by a semi-colon (;). For example, typing the following will enter 2 instructions: NOCON and ADD_DINT.

NOCON MyBOOL; ADD_DINT MyDINT OtherDINT OutputDINT

Entering a local variable
When entering operands, specify that a variable has local scope by appending a ,L after it. For example, typing “ NOCON MyLocalBOOL,L “ yields the following LD logic with a locally scoped variable as an operand:

![Figure 3 - 7. Keyboard Method Example – New Local Variable](image)

Creating New Variables
When entering operands, you may create new variables that are mapped to PLC memory by appending reference addresses. For example, typing ” NOCON %M1 “ yields the following LD logic and creates the new BOOL variable 'M00001' that is mapped to the reference address:

![Figure 4-11. Keyboard Method Example – New Variable](image)

When entering an operand, you may create a uniquely named variable and map it to a reference address by specifying a reference address. For example, typing “ NOCON MyMappedBOOL,%M1 “ yields the following LD logic and creates a new BOOL variable named 'MyMappedBOOL' that is mapped to the %M00001. You must use no space between the comma and the address.

TIP!
Reference addresses can be entered in various formats. For example %M00001, 1M, and %0001M are all valid and equivalent. Leading zeroes can be left out.

![Figure 4-12. Keyboard Method Example – New Mapped Variable](image)
Special Methods to Insert Horizontal and Vertical Wires

The methods described here are specific to inserting horizontal (H_WIRE) and vertical (V_WIRE) wires used to connect ladder logic elements. You can also insert wires by using any of the general methods used for inserting any instruction.

**Mouse Method**
1. In the LD editor, right-click the end point of an existing wire or operand terminal on a function (the starting point of your new wire).
2. Drag the mouse to the destination end point or operand terminal.
3. Release the right mouse button.
   The starting point and the end point are wired together.

**Toolbar Method**
1. On the Ladder Instruction toolbar, click the Horizontal/Vertical Wire button.
2. In the LD editor, click a cell that you want a wire in. The orientation of the wire placed depends on the orientation of the line in the mouse pointer when you click.

**Keyboard Method**
Inserting a wire in an empty cell:
1. In the LD editor, click the empty cell in which you want to place a wire.
2. Press CTRL + ARROW KEY to place the wire.
   The orientation of the wire depends on the arrow key pressed.
3. Repeat the keystrokes to continue adding or extending wires.

**TIP!**
As you move the mouse pointer about the LD logic, the line in the pointer changes to indicate the type of wire that will be inserted.

**Ladder Editor Options**
The following properties specify how ladder logic appears (online and offline) and how editing operations are carried out.

**To edit ladder logic options:**
1. In the Navigator, click the Options tab.
2. Expand the Editors folder, then the Ladder folder.
3. Right-click an options page (for example Confirmations) then select Properties from the Menu. The properties display in the Inspector window.

4. Modify the properties in the Inspector.

To restore any properties to their default values:

1. In the Navigator, click the Options tab.
2. Expand the Editor folder, then the Ladder folder and right-click one of the pages.
3. Select Reset from the Menu.

The Ladder editor properties of the page you selected are reset to their default values. The Ladder editor properties of the other options pages are left alone.

Confirmations
These options specify which ladder editor operations require a confirmation (that is, a message box appears each time the operation is performed).

Editing
These options specify the behavior of the Ladder Editor when inserting rungs of instructions.

Font and Colors
These options specify the typeface, the size and color of various items in a ladder program (online and offline).

View
These options specify which items are displayed in a ladder program.

Application Variables
Application variables are used to represent information in the controller’s logic program. These variables can use assigned to actual real-world I/O points, analog data, or represent internal program data.

Variables
A variable is a named storage location that represents a memory location in the target Micro 64 CPU. Variables may represent a simple scalar value, multi-dimensional arrays, or data structures.

TIP!

The Variable List can be displayed in a Spreadsheet view by clicking the spreadsheet icon.
Variable Data Types

The kinds of values a variable can store depend on its data type. For example, variables with a UINT data type store unsigned whole numbers. Micro 64 controllers support the following data types.

<table>
<thead>
<tr>
<th>Type</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BOOL</td>
<td>Boolean</td>
<td>The smallest unit of memory that has two states, 1 or 0.</td>
</tr>
<tr>
<td>BYTE</td>
<td>Byte</td>
<td>Has an 8-bit value with 256 values (0–255).</td>
</tr>
<tr>
<td>WORD</td>
<td>Word</td>
<td>Uses 16 consecutive bits of data memory. The valid range of word values is 0000 hex to FFFF hex.</td>
</tr>
<tr>
<td>DWORD</td>
<td>Double Word</td>
<td>Has the same characteristics as a single word data type, except that it uses 32 consecutive bits in data memory instead of only 16 bits.</td>
</tr>
<tr>
<td>UINT</td>
<td>Unsigned Integer</td>
<td>Uses 16-bit memory data locations. They have a valid range of 0 to +65535 (FFFF hex).</td>
</tr>
<tr>
<td>INT</td>
<td>Signed Integer</td>
<td>Uses 16-bit memory data locations, and are represented in 2’s complement notation. The valid range of an INT data type is –32768 to +32767.</td>
</tr>
<tr>
<td>DINT</td>
<td>Double Precision Integer</td>
<td>Stored in 32-bit data memory locations (two consecutive 16-bit memory locations). Always signed values (bit 32 is the sign bit). The valid range of a DINT data type is -2147483648 to +2147483647</td>
</tr>
<tr>
<td>REAL</td>
<td>Floating Point</td>
<td>Uses 32 consecutive bits (two consecutive 16-bit memory locations). The range of numbers that can be stored in this format is from \pm 1.401298E-45 to \pm3.402823E+38.</td>
</tr>
<tr>
<td>STRING</td>
<td>STRING</td>
<td>Eight-bit encoded characters. A single word reference is required to make two (packed) ASCII characters. The first character of the pair corresponds to the low byte of the reference word.</td>
</tr>
</tbody>
</table>

Table 3 - 2. Data Types

In Proficy Machine Edition, all variables in a project are displayed in the Variables tab of the Navigator. You create, edit, and delete variables in the Variables tab. The data type and other properties of a variable, such as reference address are configured in the Inspector.

Reference Memory

The CPU stores program data in bit memory and word reference memory tables. Memory locations are identified using alphanumeric identifiers called references. The reference’s letter prefix identifies the memory area. The numerical value is the offset within that memory area, for example %AQ0056 represents element 56 in the Analog Output (AQ) table.
Word (Register) References

<table>
<thead>
<tr>
<th>Type</th>
<th>Memory Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%AI</td>
<td>%AI001 to %AI0128</td>
<td>The prefix %AI represents an analog input register. An analog input register holds the value of one analog input or other non-discrete value.</td>
</tr>
<tr>
<td>%AQ</td>
<td>%AQ001 to %AQ0128</td>
<td>The prefix %AQ represents an analog output register. An analog output register holds the value of one analog output or other non-discrete value.</td>
</tr>
<tr>
<td>%R</td>
<td>%R0001 to %R032640</td>
<td>The prefix %R represents system register references that can store program data such as the results of calculations.</td>
</tr>
</tbody>
</table>

Table 3 - 3. Word (Register) References

Bit (Discrete) References

<table>
<thead>
<tr>
<th>Type</th>
<th>Memory Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>%I</td>
<td>%I001 to %I0512</td>
<td>Represents input references. %I references are located in the input status table, which stores the state of all inputs received from input modules during the last input scan. A reference address is assigned to discrete input modules using your programming software. %I memory is always retentive.</td>
</tr>
<tr>
<td>%Q</td>
<td>%Q001 to %Q0512</td>
<td>Represents output references. The coil check function checks for multiple uses of %Q references with relay coils or outputs on functions. %Q references are located in the output status table, which stores the state of the output references as last set by the application program. This output status table’s values are sent to output modules at the end of the program scan. A reference address is assigned to discrete output modules using your programming software. A particular %Q reference may be either retentive or non-retentive.</td>
</tr>
<tr>
<td>%M</td>
<td>%M001 to %M01024</td>
<td>Represents internal references. The coil check function of your programming software checks for multiple uses of %M references with relay coils or outputs on functions. A particular %M reference may be either retentive or non-retentive.</td>
</tr>
<tr>
<td>%T</td>
<td>%T001 to %T0256</td>
<td>Represents temporary references. These references are never checked for multiple coil use and can be used many times in the same program. Because this memory is intended for temporary use, it is cleared on Stop-to-Run transitions and cannot be used with retentive coils.</td>
</tr>
<tr>
<td>%S</td>
<td>32 bits each</td>
<td>Represent system status references. These references are used to access special CPU data such as timers, scan information, and fault information. For example, the %SC0012 bit can be used to check the status of the CPU fault table.</td>
</tr>
<tr>
<td>%G</td>
<td>%G001 to %G01280</td>
<td>Represents global data references. These references are used to access data shared among several control systems.</td>
</tr>
</tbody>
</table>

Table 3 - 4. Bit (Discrete) References
System Status References

System status references in the CPU are assigned to %S, %SA, %SB, and %SC memory. The four timed contacts include #T_10MS, #T_100MS, #T_SEC, and #T_MIN. Examples of other system status references include #FST_SCN (First Scan), #ALW_ON (Always On), and #ALW_OFF (Always Off).

<table>
<thead>
<tr>
<th>Reference</th>
<th>Nickname</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>%S0001</td>
<td>FST_SCN</td>
<td>Set to 1 when the current sweep is the first sweep.</td>
</tr>
<tr>
<td>%S0002</td>
<td>LST_SCN</td>
<td>Reset from 1 to 0 when the current sweep is the last sweep.</td>
</tr>
<tr>
<td>%S0003</td>
<td>T_10MS</td>
<td>0.01 second timer contact.</td>
</tr>
<tr>
<td>%S0004</td>
<td>T_100MS</td>
<td>0.1 second timer contact.</td>
</tr>
<tr>
<td>%S0005</td>
<td>T_SEC</td>
<td>1.0 second timer contact.</td>
</tr>
<tr>
<td>%S0006</td>
<td>T_MIN</td>
<td>1.0 minute timer contact.</td>
</tr>
<tr>
<td>%S0007</td>
<td>ALW_ON</td>
<td>Always ON.</td>
</tr>
<tr>
<td>%S0008</td>
<td>ALW_OFF</td>
<td>Always OFF.</td>
</tr>
<tr>
<td>%S0009</td>
<td>SY_FULL</td>
<td>Set when the PLC fault table fills up. Cleared when an entry is removed and when the PLC fault table is cleared.</td>
</tr>
<tr>
<td>%S0010</td>
<td>IO_FULL</td>
<td>Set when the I/O fault table fills up. Cleared when an entry is removed from the I/O fault table and when the I/O fault table is cleared.</td>
</tr>
<tr>
<td>%S0011</td>
<td>OVR_PRE</td>
<td>Set when an override exists in %I, %Q, %M, or %G memory.</td>
</tr>
<tr>
<td>%S0012</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>%S0013</td>
<td>PRG_CHK</td>
<td>Set when background program check is active. Cleared when the background program check is inactive.</td>
</tr>
<tr>
<td>%S0014</td>
<td>PLC_BAT</td>
<td>Set to indicate a bad battery in the CPU. The contact reference is updated once per sweep. This bit is supported by Micro 23, Micro 28 and Micro 64 PLCs only.</td>
</tr>
<tr>
<td>%S0015, 16</td>
<td></td>
<td>Reserved</td>
</tr>
<tr>
<td>%S0017</td>
<td>SNPXACT</td>
<td>SNP-X host is actively attached to CPU port 1. (Port 2 defaults to disabled, and must be activated with a CRQ).</td>
</tr>
<tr>
<td>%S0018</td>
<td>SNPX_RD</td>
<td>SNP-X host has read data from CPU port 1.</td>
</tr>
<tr>
<td>%S0019</td>
<td>SNPX WT</td>
<td>SNP-X host has written data to CPU port 1</td>
</tr>
<tr>
<td>%S0020</td>
<td>FF_OVR</td>
<td>Set ON when a relational function using REAL data executes successfully. It is cleared when either input is NaN (Not a Number).</td>
</tr>
<tr>
<td>%S0021</td>
<td>USR_SW</td>
<td>Set to report a Fatal Fault Override.</td>
</tr>
<tr>
<td>%S0022</td>
<td></td>
<td>Set to reflect the state of the CPU mode switch.</td>
</tr>
<tr>
<td></td>
<td>1 = Run/On 0 = Stop/Off</td>
<td></td>
</tr>
<tr>
<td>%S0023-32</td>
<td>reserved</td>
<td></td>
</tr>
<tr>
<td>%SA0001</td>
<td>PB_SUM</td>
<td>Set when a checksum calculated on the application program does not match the reference checksum. If the fault was due to a temporary failure, the discrete bit can be cleared by again storing the program to the Micro PLC. If the fault was due to a hard RAM failure, the Micro PLC must be replaced.</td>
</tr>
<tr>
<td>%SA0002</td>
<td>OV_SWP</td>
<td>Set when a Micro PLC in CONSTANT SWEEP mode detects that the previous sweep took longer than the time specified. Cleared when the PLC detects that the previous sweep did not take longer than specified. Also cleared during transition from STOP to RUN mode.</td>
</tr>
<tr>
<td>%SA0003</td>
<td>APL_FLT</td>
<td>Set when an application fault occurs. Cleared when the Micro PLC transitions from STOP to RUN mode.</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>---------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>%SA0004-8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>%SA0009</td>
<td>_CFG_MM</td>
<td>Set when a configuration mismatch is detected during powerup or a configuration store. Cleared by powering up the Micro PLC after correcting the condition.</td>
</tr>
<tr>
<td>%SA0010</td>
<td>HRD_CPU</td>
<td>Set when the diagnostics detects a problem with the Micro PLC hardware. Requires replacing the Micro PLC. This bit is supported by 28-point Micro PLCs only.</td>
</tr>
<tr>
<td>%SA0011</td>
<td>LOW_BAT</td>
<td>Set when a low battery fault occurs. Cleared by replacing the battery then powering up the Micro PLC. This bit is supported by Micro 23, Micro 28 and Micro 64 PLCs only.</td>
</tr>
<tr>
<td>%SA0012,13</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>%SA0014</td>
<td>LOS_IOM</td>
<td>Set when an expansion module stops communicating with the CPU. Cleared by replacing the module and cycling system power.</td>
</tr>
<tr>
<td>%SA0015-18</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>%SA0019</td>
<td>ADD_IOM</td>
<td>Set when an expansion module is added. Cleared by cycling PLC power and when the configuration matches the hardware after a store.</td>
</tr>
<tr>
<td>%SA0020-31</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>%SB0001-8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>%SB0009</td>
<td>NO_PROG</td>
<td>Set when an attempt is made to put the PLC in Run mode when there is no executable application program stored in the CPU. Cleared by storing an application program to the CPU and putting the PLC in Run mode.</td>
</tr>
<tr>
<td>%SB0010</td>
<td>BAD_RAM</td>
<td>Set when the Micro PLC detects corrupted RAM memory at powerup. Cleared when RAM memory is valid at powerup.</td>
</tr>
<tr>
<td>%SB0011</td>
<td>BAD_PWD</td>
<td>Set when a password access violation occurs. Cleared when the PLC fault table is cleared.</td>
</tr>
<tr>
<td>%SB0012</td>
<td>SFT_CPU</td>
<td>Set when the Micro PLC detects an unrecoverable error in the software. Cleared by clearing the PLC fault table.</td>
</tr>
<tr>
<td>%SB0013</td>
<td>STOR_ER</td>
<td>Set when an error occurs during a programmer store operation. Cleared when a store operation is completed successfully.</td>
</tr>
<tr>
<td>%SB0014</td>
<td>SY_PRES</td>
<td>Set as long as there is at least one entry in the PLC fault table. Cleared when the PLC fault table has no entries.</td>
</tr>
<tr>
<td>%SC0001-8</td>
<td>Reserved</td>
<td></td>
</tr>
<tr>
<td>%SC0009</td>
<td>ANY_FLT</td>
<td>Set when any fault occurs. Cleared when both fault tables are cleared.</td>
</tr>
<tr>
<td>.%SC0010</td>
<td>SY_FLT</td>
<td>Set when any fault occurs that causes an entry to be placed in the PLC fault table. Cleared when the PLC fault table is cleared.</td>
</tr>
<tr>
<td>.%SC0011</td>
<td>IO_FLT</td>
<td>Set when any fault occurs that causes an entry to be placed in the I/O fault table. Cleared when the I/O fault table is cleared.</td>
</tr>
<tr>
<td>.%SC0012</td>
<td>SY_PRES</td>
<td>Set as long as there is at least one entry in the PLC fault table. Cleared when the PLC fault table has no entries.</td>
</tr>
<tr>
<td>.%SC0013</td>
<td>IO_PRES</td>
<td>Set as long as there is at least one entry in the I/O fault table. Cleared when the I/O fault table has no entries.</td>
</tr>
<tr>
<td>.%SC0014</td>
<td>HRD_FLT</td>
<td>Set when a hardware fault occurs. Cleared when both fault tables have no entries. This bit is supported by Micro 23, Micro 28, and Micro 64 PLCs only.</td>
</tr>
<tr>
<td>.%SC0015</td>
<td>SFT_FLT</td>
<td>Set when a software fault occurs. Cleared when both fault tables have no entries.</td>
</tr>
</tbody>
</table>

**Table 3 - 5. System Status References**
Defining Application Variables

Application variables are defined for a target using the Variable tab of the Navigator. To define a new Variable, first select the Variable tab of the Navigator. Then select the main Variable node New Variable menu item. After you select the Variable data type, you may edit the new Variable’s name.

A Variable’s name may consist of up to 32 characters. A variable also has an associated description that may consist of up to 255 characters.

A variable’s properties are displayed in the Inspector when the Variable is selected anywhere in Machine Edition. You can also edit the Variable’s information in the Inspector.
Validating Logic Applications

The validation process checks the target application for errors. Any messages or errors are displayed in the Build tab of the Feedback Zone. Each item in a target has its own validation checklist for potential errors. Among other things, Machine Edition ensures that properties for items are within allowable ranges; that all items referenced actually exist.

To validate a target, in the Project tab of the Navigator, right-click the target and choose Validate -or- click the Validate toolbar button. Any errors or warnings found during the validation appear in the Build tab of the Feedback Zone.

- For help on an error or warning, select it in the Build tab and help will appear in the Companion (if it’s open).
- Press F4 to cycle through errors and warnings, locating each error in the project in turn.

TIP!
Double-click an error or warning to jump to its location.

Downloading Logic Applications

Once your application has been validated with no Errors, you can download and run the application in the controller. To validate, download, and start the application all at once, in the Project tab of the Navigator right-click the controller target on which you want to execute logic, and then select the Download and Start toolbar button or menu item.

Depending on the state of the controller, Logic Developer PLC will display a download dialog so that you can select which components of the application to download to the controller. An example of this dialog is shown in Figure 4-18.

After the application has been successfully downloaded to the controller, the Equality Status of the Project with the controller is displayed on the Machine Edition status bar as Config EQ/NE, Logic EQ/NE for Equal and Not Equal.

NOTE
It is recommended that when downloading, the box Write ALL items to flash memory should be checked.
Monitoring Logic Applications

Once your application has been downloaded to the controller, the logic execution and application data may be monitored in a number of ways:

- Logic information may be displayed live in the LD and Instruction List Editors
- Application variables may be displayed live in the Data Watch window
- Variables may be trended in the Data Monitor
- Application reference memory may be displayed in the Reference View Tables

Note that Logic Developer PLC has a concept of Logic Equality that will determine which of these data monitoring tools are available to the user. Logic Developer PLC will only animate the Logic and Application Variables if Logic Equality exists between the controller and the project. The Logic Equity status is indicated in the LD Editor window as well as on the Machine Edition status bar.

Monitoring Ladder Diagram Logic

An example of monitoring Ladder Diagram logic is shown in Figure 3-13. Please note that the highlighting of the Boolean logic elements indicates the state of the discrete variables assigned to the logic elements. If the discrete variable has been forced, the variable is displayed in red and underlined, such as the REGP_DONE_SCAN variable at the end of Rung 10. Values of internal variables are also displayed above the Variable Name in the logic.

Logic equality with the controller is indicated by the yellow thunderbolts in the corners of the Ladder Diagram editor.

![Figure 3 - 13. Ladder Diagram Logic Monitoring](image-url)
Data Watch

The Data Watch tool is a debugging tool that enables you to monitor the values and forces of variables. With the Data Watch tool, you can monitor individual variables or user-defined lists of variables. Watch lists can be imported, exported, or saved with a project.

**TIP!**

The Data Watch displays forced variables with a red underline.

There are three tabs in the Data Watch tool:

- **The Static** tab contains all variables that you add to the Data Watch tool yourself.
- **The Auto** tab contains variables currently selected in the Variable List or associated with the currently selected instruction in the logic.
- **The Watch List** tab contains all variables in the currently selected watch list. A watch list lets you create and save a list of variables to monitor. You can define one or more watch list(s); however, you can monitor only one watch list in the Data Watch tool at a time.

Values of variables in the Data Watch tool are displayed in the **Value** column. To change a value's display format, right-click the valve cell and select the desired data format.

![Figure 3 - 14. Data Watch Example](image)

**TIP!**

When you add variables to a watch list by manually entering variable names, Machine Edition can also identify variables by their Addresses.
Data Monitor

The Data Monitor utility displays recent values of variables in a trend graph as they change over time.

To open the Data Monitor utility window, double-click the Data Monitor in the Utilities tab of the Navigator.

You can drag variables directly from the Variable List onto the Data Monitor window. A trend line begins for the variable immediately. The Data Monitor utility can chart values of integer, real, and discrete variables.

The Data Monitor window looks something like this:

Figure 3 - 15. Data Monitor Example

TIP!

Variables in the Trend Legend are drawn in the order they appear, with the top variable drawn last. If a particular trend line is obscured by others, double-click the variable to move it to the top of the list.
Reference View Table

The Reference View Tables display the memory tables of the controller. For convenience, Logic Develop PLC provides a set of default reference view tables that may be displayed by opening the desired reference table from the Navigator.

You may also create and name your own reference view tables. In these tables you can mix different memory types, as you like. You can also use the right mouse menu to modify the data display format for a specific memory location, or for the entire table. An example of a reference view table for the %R register table is as follows:

![Reference View Table Example](image)

**Figure 3 - 16. Reference View Table Example**

**TIP!**

If you move the mouse over a data cell, the Tooltip will popup to display the Variable Name and Variable Description of the variable assigned to the memory location.

Dynamic Cross References

The dynamic cross-reference feature of Machine Edition is a very useful tool in debugging logic applications. This function will list everywhere in a Project where an item has been referenced (used). By selecting the References tab of the Feedback Zone, it will contain a list of the places in the project where a selected item (variable) is used. In Logic Developer - PLC, the References tab may also contain a list of all the Calls to a given block of logic.

With the References tab active, click a variable in the Variables tab of the Navigator to see a list of the places where it is used. Double-click an entry in the References tab to jump to the place in the project where that reference occurs. You may also cycle through those entries using the F4 key.
With the References tab active, click one of the following:

- A CALL instruction in an block
- A block in a Called Blocks folder
- A block in the Logic folder

The References tab in the Feedback Zone will be cleared and will display the location of all calls to the block called by the CALL instruction for the block selected. This applies regardless of the language of the called block.

Double-click an entry in the References tab to jump to the location in the block where that CALL is made.

An example of the feedback zone References tab is shown below.

Figure 3-17. Feedback Zone References Example
Lab Exercise - Programming the Micro 64 Controller

Lab objectives: At the end of this exercise, the student will be able to:

- Create application logic and application variables
- Download applications to the Micro 64 controller
- Monitor applications in the Micro 64 controller

Working with Micro 64 Programs

Create a simple ladder program using various instructions and assign variables to the logic.

Opening a Ladder Program

1. In the Project tab of the Navigator, under the Logic folder, right click on the _MAIN block node, then choose Open from the menu or double click on the _MAIN block node.

The ladder logic editor for the _MAIN block is opened, ready for editing.

Inserting Instructions - Basic Method

1. In the Ladder Editor, right-click the first cell in the first rung, and select the Place Instruction menu item, or click the cell the new instruction will be in and begin typing the instruction.
In either case, a smart list appears prompting you to enter an instruction.

2. Type or select from the list **NOCON** for a normally open contact.

   **Normally Open Contact- NOCON** - A normally open contact acts as a switch that passes power flow if the associated parameter is **ON (1)**.

   **NOTE**

   You do not need to type in the entire mnemonic. As soon as the desired mnemonic is highlighted in the smart list, you can press the TAB key or the Enter key to insert it.

3. Press the **ENTER** key.

4. The Variable dialog box will appear. Select from the list or type the variable name. Type **STARTPB**.

   If the text in the selection box appears:

   - **Red**: the entered item is invalid and the Smart List will not accept it.

   - **Grey**: the entered item is valid, but might cause problems.

   - **Green**: the entered item does not appear in the list. However, you can select it and Machine Edition will create a new item with the given name.
Black: the item appears in the list and is a valid selection.

5. The instruction appears in the Ladder Editor with its associated application variable.

Toolbar Method of Inserting LD Instructions

1. Locate the Ladder Instructions toolbar.

If the Ladder instructions toolbar isn’t visible, click on the Tools pull-down menu and select Toolbars > Logic Developer - PLC.

2. The Ladder Instructions Toolbar should now be visible.

3. Click the button on the Ladder Instructions toolbar to select a normally closed contact.

Normally Closed Contact – NCCON - A normally closed contact acts as a switch that passes power flow if the associated parameter is OFF (0).

4. Now click on the cell in column 2 to drop the contact.
5. Press the ENTER key. The Variable dialog box will appear. Select from the list or type the variable name. Type **STOPPB**.

---

**Toolchest Method of Inserting LD Instructions**

1. Click the **Toolchest icon** on the Tools toolbar. The Toolchest window opens.

2. In the Toolchest, expand the **PLC LD Instructions** drawer. The drawer opens in the Toolchest window. Each folder in the drawer contains several closely related instructions.

3. Expand the folder named **Coils** by clicking on the + sign next to it and select the $ Coil instruction mnemonic then hold down the left mouse button and drag it to the Ladder Editor.

**Coil (COIL)**  - When power is applied to a COIL instruction the parameter turns ON (1). When no power is applied, the parameter is OFF (0). Because COIL is an output instruction, it should appear at the end of a rung.
4. Drop the Coil instruction on the rung in the cell in column 3 by releasing the left mouse button. Logic Developer PLC will automatically place the Coil in the 10th column by default.

5. Follow the same steps to name the Variable and enter a variable named MOTOR. You will need to highlight the Coil instruction and press ENTER to assign the application variable to the instruction.

![Image of Logic Developer PLC interface]

Insert Instructions - Keyboard Method

1. To insert a new rung, click on the cell below the STARTPB contact.

2. Type instruction mnemonics followed by parameters, separated by semicolons (;). A Smart list appears as soon as you begin typing.

3. Add a normally open contact with a $ Hundreds Timer and a Coil.

4. Type NOCON MOTOR; TMR_HUNDS DWELL; COIL TIMER_DONE

![Smart list of mnemonics and parameters]

Timer Hundreds (TMR_HUNDS)
The Timer Hundreds (TMR_HUNDS) instruction increments every hundredth of a second while it receives power flow, and resets to 0 when power flow stops. The timer passes power after the specified interval "PV" has elapsed, as long as power

5. Press ENTER. The instructions and parameters specified by the mnemonic string are inserted on the current rung.

The following logic will appear.

![Image of Logic Developer PLC interface with inserted instructions]

To find out more information about the TMR_HUND instruction, click on it and press F1. The InfoViewer will display the complete detailed help for this instruction.
6. Assign the number of milliseconds to the Preset Value Time. Double-click the cell next to the PV parameter and type in 900 (equals 900 hundredths which equals 9 seconds).

7. Add another normally closed contact instruction – to rung 1 after the STOPPB contact. Label it TIMER_DONE.

To Insert A Branch
1. Select a cell in Rung 2 and then select the Insert Row menu, or press the Ctrl+R keys. Click on the power rail to the left of the empty cell in column 1. Click the right-mouse button and drag towards the end point, just after the STOPPB contact. The pointer will appear as ☒ with a line connecting it to the starting point.
2. Position the pointer at the desired end point and release the mouse button when the pointer appears normal.
3. A branch appears between the start and end points.

Duplicate Instructions
1. In the Ladder Editor, select an instruction (Motor contact in Rung 2) press CTRL then click and drag the selection to a new location (in the branch). The mouse pointer will appear as a while a copy operation is pending. A fuzzy box indicates where the duplicate instruction will appear when dropped.
To Move An Instruction

1. Select an instruction (STOPPB contact) to be moved. Drag the selection to a new location in column 4. The mouse pointer will appear as a while a move operation is pending. A fuzzy box indicates where the instruction will appear when dropped. Once the contact is moved, attach the first contact to the horizontal line in Rung 1 using the Horizontal /Vertical Wire tool.

Download and Start a Project

The Download process creates (or "builds") all of the runtime files necessary for a target to perform its role in a completed application. Then, it downloads those files to the target device. Before creating the runtime files, Machine Edition also performs a validation.

To download to the Micro 64 controller, you must go Online with the target usually by connecting to the CPU through a network or one of your PC’s communication ports. If you are going to connect to the controller over Ethernet, then you will need to setup the IP Address of the CPU in order to be able to communicate with it. Machine Edition has a Set Temporary IP Address Utility that is used to initially setup a controller.

Set Temporary IP Address Utility

The Set Temporary IP Address (“Set IP”) utility lets you temporarily specify the IP Address of certain types of devices or PLCs connected to the local computer over the network. The Set IP utility is most useful during the initial configuration of PLCs and devices with no serial port.

The following restrictions apply when using the Set Temporary IP Address utility for a GE Fanuc Micro 64 CPU or Ethernet module:

- The Set Temporary IP Address utility does not function if communications with the networked Micro 64 travel through a router. Communications through switches and hubs are fine.
- The current user logged on the development computer must have full administrator privileges.
• The target Micro 64 must be located on the development computer's local subnet, as specified by the computer's subnet mask and the IP addresses of the computer and the Micro 64.

**Warning!**
The IP address set by the Set Temporary IP Address utility does not survive a power cycle.

After setting the temporary IP address and before the power cycles, you can create a Hardware Configuration for the GE Fanuc Micro 64 controller to set the target's IP Address property to the desired permanent IP address. After you configure the IP Address, you download the Hardware Configuration to the controller to make this a permanent setting.

**To Set a Temporary IP Address on a Device or PLC**

1. Connect the PC Ethernet cable the Ethernet port on the Micro 64.
2. Do one of the following:
   • In the Utilities tab of the Navigator, double-click the Set Temporary IP Address node (or right-click it and choose Open). - or -
   • Right-click the target, point to Offline Commands, and choose Set Temporary IP Address.

   The Set Temporary IP Address dialog box appears.

3. In the Set Temporary IP Address dialog box, do all of the following:
   a. In **MAC Address**, specify the MAC address for the target. This is the MAC address of the Micro 64 controller Ethernet port. The MAC address is located on the top left side of the Micro 64 demo.
   b. In **IP Address to set**, specify the temporary IP address you want to set on the target. (10.10.0.2)
   c. If necessary, in **Network Interface Selection**, select the check box and specify the network interface on which the target is located.
4. When the rest of the fields are properly configured, click the **Set IP** button.
The IP Address of the specified target is set to the indicated address.
This may take up to a minute.

5. The Micro 64 Ethernet has the IP address already preloaded in its firmware and storing of the IP address will not be required for this lab.

Going Online With a Controller

To specific connection to the target go to the Project tab of the Navigator, click on the target. In the Inspector window select Physical Port Ethernet and set the IP address as 10.10.0.2
To go online to a specific target, in the **Project** tab of the Navigator, right click on the target to which you want to download the runtime files to and select **Go Online**. Alternately, you can select the **Go Online** icon from the online toolbar.
The Target icon in the navigator and the Machine Edition Status Bar indicates the online status with the controller.

**Downloading a Project to a Controller**

1. To download to a specific target, in the Project tab of the Navigator, right click on the target to which you want to download the runtime files to and select **Download to PLC**. Alternately, you can select the **Download Active Target** icon from the online toolbar.

For each target, Machine Edition performs a validation, builds the runtime files, and attempts to download the application to the target device.

Any errors that occur are displayed in the Build tab of the Feedback Zone.
Placing the Controller in Run Mode

To place the controller in Run Mode, in the **Project** tab of the Navigator, right click on the target and select **Online Commands > Start PLC**. Alternately, you can select the **Start Active Target** icon from the online toolbar.

**Viewing Logic Online**

1. Double click on the _MAIN Ladder block to open the Ladder Editor Window. You should now see that the logic is online (green lines and lighting bolts).

2. The Ladder Editor allows you to test the logic. Right click on the **STARTPB** variable and select **Turn On** from the menu.

3. Immediately turn **STARTPB** off to simulate a momentary switch.

4. You will notice the Timer has started - when the timer hits 900 the motor will turn off.
5. Select other instructions - right click and select Turn On/Off or Force On/Off and see what happens.
Review

In this module, you have learned to:

- Identify a ladder program and what it contains
- Configure the Ladder Diagram Editor options
- Identify each of the Ladder instruction groups and the instructions in those groups
- Create programs using various instructions
- Assign variables to the program elements
- Go Online with the controller, Download the project to the controller, Start the controller