HMI

Human Machine Interface

Operating Instructions

HMI

Human Machine Interface

Operating Instructions – Addendum

Enhanced capabilities now available through HMI Screen Builder software and which are not covered in this manual are summarized below. Each new feature description includes details on implementation and use. It is highly recommended that new HMI users first read the instruction manual to become familiar with its set up and use.

• *Registers no longer require a default value. By not entering a default value, the HMI will save the user-entered value which can be updated.*
Setting up a register in earlier versions of Screen Builder required a value of 1-8 numeric characters. In the new version of Screen Builder, a register can accept no entry at all or 1-8 numeric characters. This new feature allows a user to dynamically enter a value without having a default value of 0 forced into the register.

• *Option in Screen Builder to turn on back light.*
After completing your project, Screen Builder creates the code that is used for the MicroLynx or Lynx (see page 28 of the HMI Manual). Two new subroutines are added to the Screen Builder code which can be used to turn the HMI's back light on or off. In your MicroLynx or Lynx program, simply call either subroutine to turn the back light on or off (subroutine names are BL_ON, BL_OFF).

• *Screen Builder will generate two new subroutines: one for initializing registers and one for updating the value of the register from the HMI to the LYNX or MicroLYNX after cycling power.*
Initializing registers to a default value can be accomplished using the IREG subroutines. To update the register value from the HMI to the MicroLYNX, simply use the RREG subroutines. The RREG feature is used to restore the last entered register values to the MicroLYNX after cycling power. Within your program, simply call each individual subroutine or update all registers at once using a single call statement in your program. Updating your register values from the MicroLYNX to the HMI, or HMI to MicroLYNX, can now be accomplished easily.

• *A new Compile view window allows users to view the compiled code downloaded to the HMI. This is a great tool for advanced users to modify their HMI project.*
After completing and saving your Screen Builder project, the Compile view window can be opened by going to the View tab in the Screen Builder window. In the drop down list, click on the Compile view tab and the generated escape codes will appear.
This new window will show the generated escape codes that will be sent to the HMI. Users can now see the address and location of each escape that was generated for the HMI (see page 54 of the HMI Manual for escape codes list). With the ability to view this information, advanced users can add their own escape codes and know which addresses are available.

Rev07.28.2003

Table of Contents

List of Figures

List of Tables

SECTION 1

The Human/Machine Interface

Section Overview

This section introduces the user to the Human/Machine Interface (HMI). Covered are:

- Introduction.
- Features and Benefits.
- Mechanical Specifications.
- Electrical Specifications.
- Environmental Specifications.
- Connection Overview.
- Pin Configuration.

Introduction

The Human/Machine Interface is a multi-function interface for the LYNX Controller and MicroLYNX. Its key descriptor is "versatility".

The HMI features three modes of operation: Programmable Display, Thumbwheel Emulation and Register. These modes can be switched on-the-fly. In addition, it has six programmable function keys, 64 registers for numeric storage, and a full 4kbytes of user storage space. The HMI may be programmed by using the included Graphic User Interface: The HMI ScreenBuilder, or by using escape codes.

These features combine to make the HMI a powerful addition to the LYNX family of machine control solutions.

Features and Benefits

- Very low cost.
- Small size 0.625" X 5.9" X 4.4" (15.9 X 149.8 X 111.7 mm).
- 3 Modes of operation: programmable display, thumbwheel emulation and register.
- Modes of operation can be switched on-the-fly.
- 64 registers for numeric storage (10 digits).
- 4 kbytes of user storage.
- User output.
- 6 programmable function keys.
- Communications via RS-485.
- Supertwist LCD Display.
- Can be used to control multiple LYNX products in party mode.
- Optional communication cables and mounting hardware for use with MicroLYNX.

Mechanical Specifications

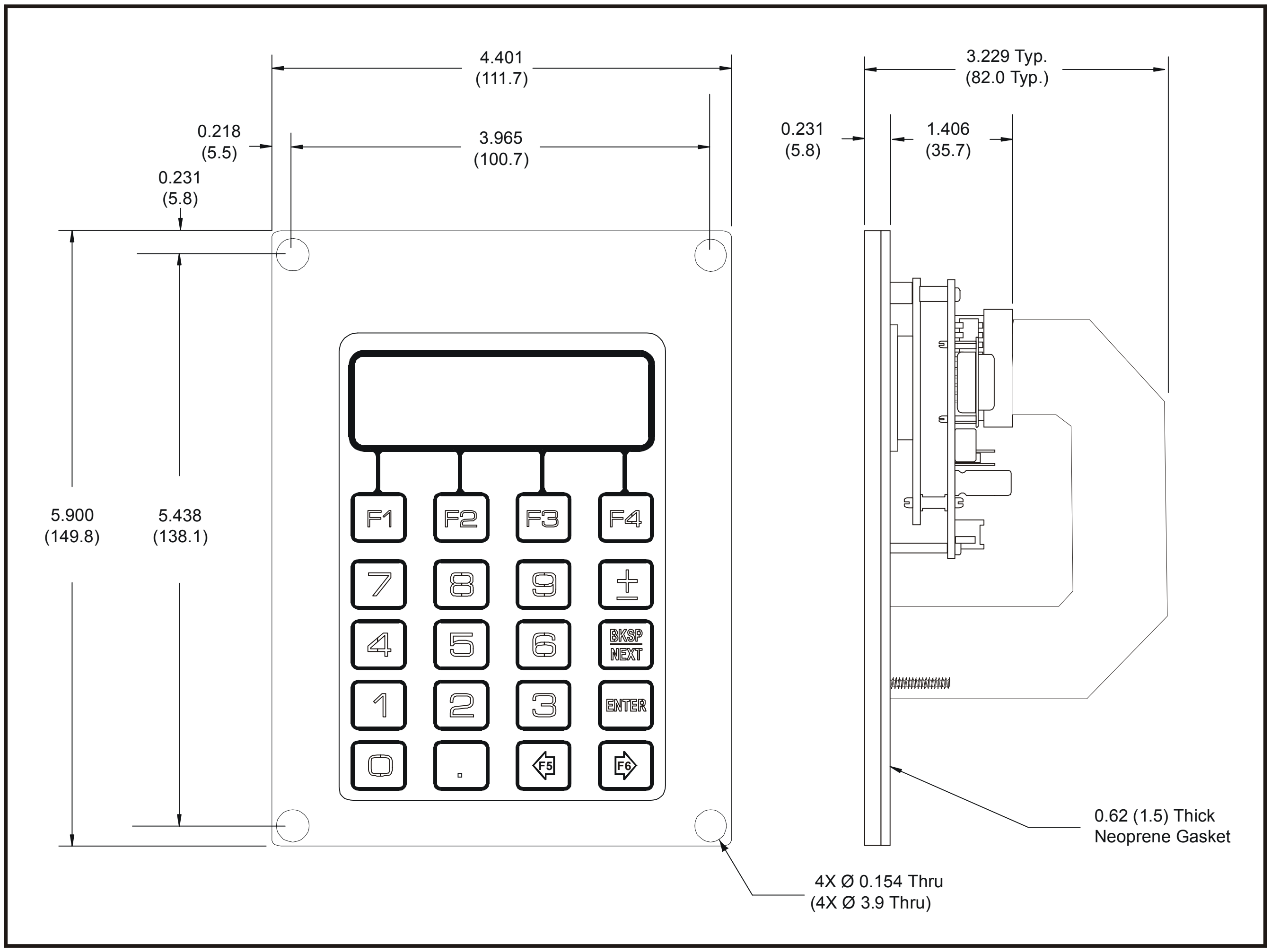


Figure 1.1: HMI Dimensional Information, Dimensions in Inches (mm)

Electrical Specifications

Power Supply Requirements

V+ .... +12 to +80VDC

Communications

Protocol.. RS-485
BAUD Rates 4800, 9600, 19.2k, 38.4k
Duplex .... Full

User Output

Maximum ratings are shown.

Voltage 200VDC
Current 500mA

Connection Overview

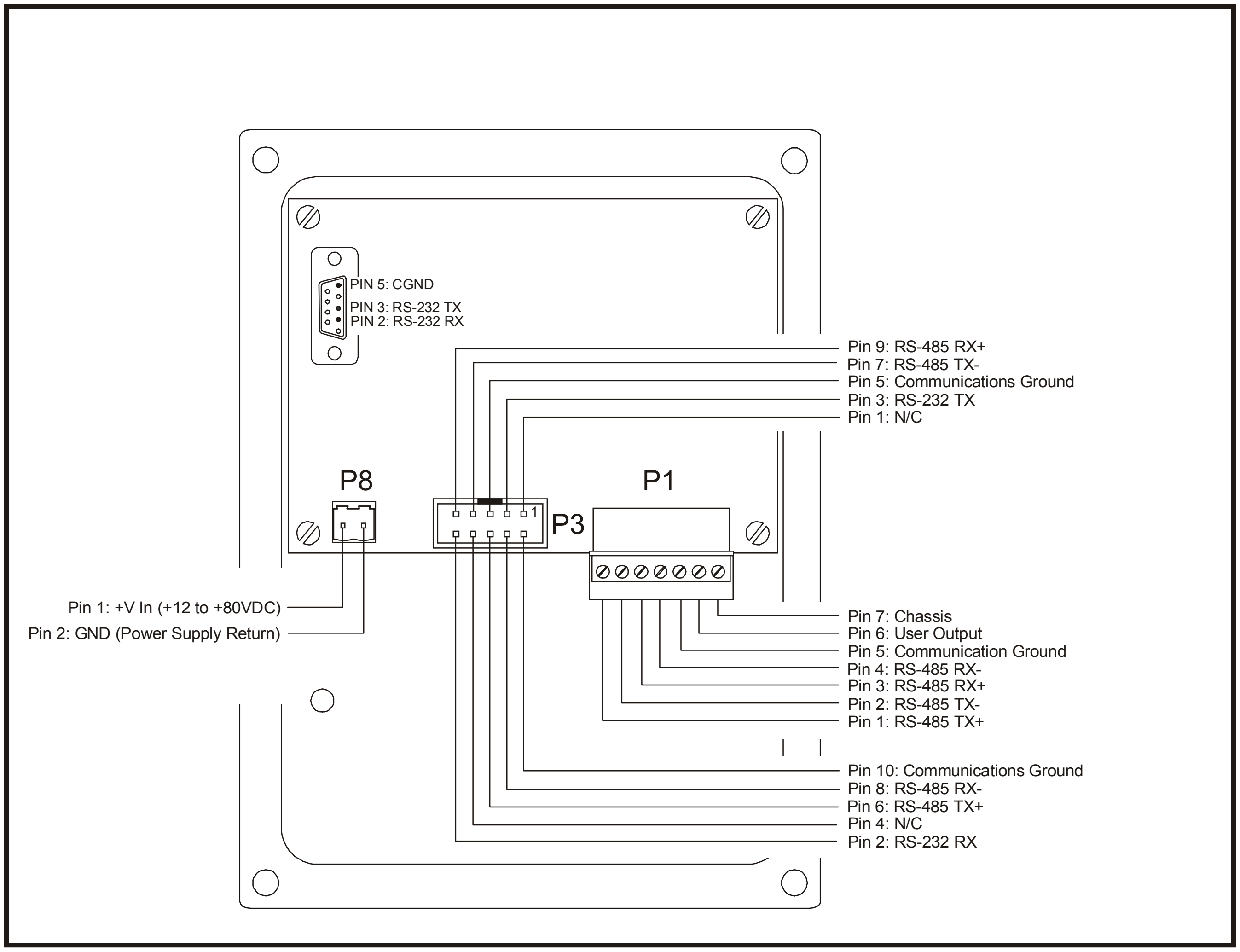


Figure 1.2: HMI Connection Overview

NOTE: The HMI does not feature on-board RS-232 communications. The DB-9 connector signals connect directly to their corresponding pins on connector P3. This connector exists to facilitate interfacing your host MicroLYNX to your PC using common cableing. Commands dircted to the HMI are forwarded through MicroLYNX RS-485.

SECTION 2

Connecting and Mounting the HMI

Section Overview

This section covers the wiring, installation and mounting of the Human/Machine Interface. This section will show the HMI used with various LYNX controller products.

- Connecting Power.
- Connecting Communications.
- Connecting the User Output.
- Mounting.

Connecting Power

The Human/Machine Interface can accept a power input ranging from +12VDC to +80VDC at connector P1. This supply would also typically be supplying +V to the LYNX controller and/or IMS driver products in your system.

If the HMI is being powered separately from the driver, a regulated +12 to +80VDC supply may be used. If the HMI and the stepping motor driver are sharing a power supply, then an unregulated supply such as the IMS ISP-200 is recommended because of the surge currents present in stepping motor systems.

See figure 1.2 in the previous section for P1 connector location.

Wiring Specifications

Wire Type	18 to 26 AWG, shielded twisted pair
Strip Length	0.20" (5.0mm)
Terminal Screw Torque	3.0 lbs-in (0.33 N-m)

WARNING! When using an unregulated supply, ensure that the output voltage does not exceed the maximum driver input voltage due to variations in line voltage! It is recommended that an input line filter be used on the power supply to limit voltage spikes to the system!

NOTE: All system wiring should be shielded twisted pair with a minimum of 1 twist per inch to reduce electrical noise! Power and motor wiring should be run separately from signal carrying wiring. Do not "daisy-chain" power wiring to system components.

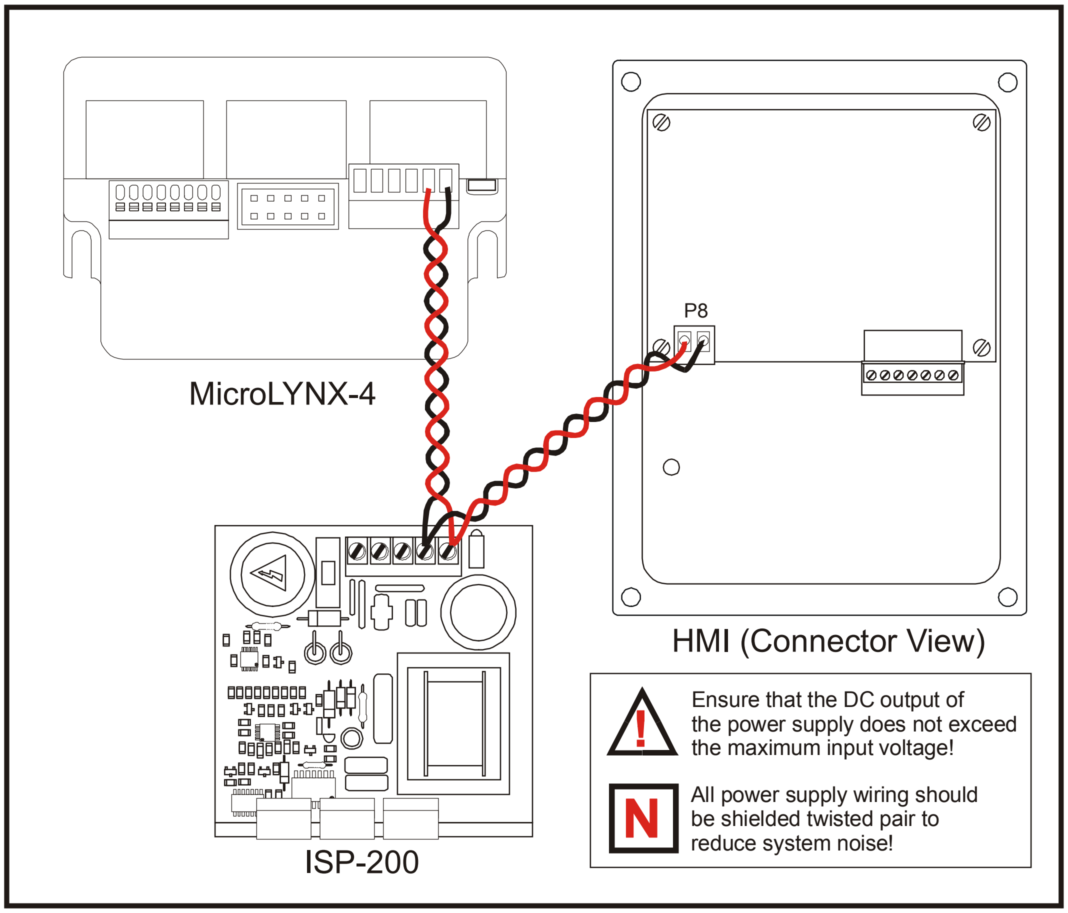

MicroLYNX-4
P8
HMI (Connector View)
ISP-200
Ensure that the DC output of the power supply does not exceed the maximum input voltage!
All power supply wiring should be shielded twisted pair to reduce system noise!

Figure 2.1: Connecting Power to the HMI

Connecting Communications

In this subsection there are three communications connection examples. The first two connection examples use the optional communication cables available from IMS. These cables are inexpensive and of quality manufacture. IMS recommends the use of these cables for prototyping and in systems where the HMI will interact with a single MicroLYNX or two axis MicroLYNX systems. Use of these cables simplify the wiring and connection of the system, and also eliminate down time resulting from wiring errors.

Connecting Communications, Single MicroLYNX

This method of connecting communications to the HMI and a single MicroLYNX system is shown using the optional communications cable MX-CC400-000. This cable is 3.5" long and will typically be used in concert with the optional mounting hardware kit, MX-CC200-001.

Communications are connected from the RS-232 port of the user's host PC to the DB-9 connector of the HMI. Please note that the signal path through this connector does not tie directly to any HMI hardware. It is a

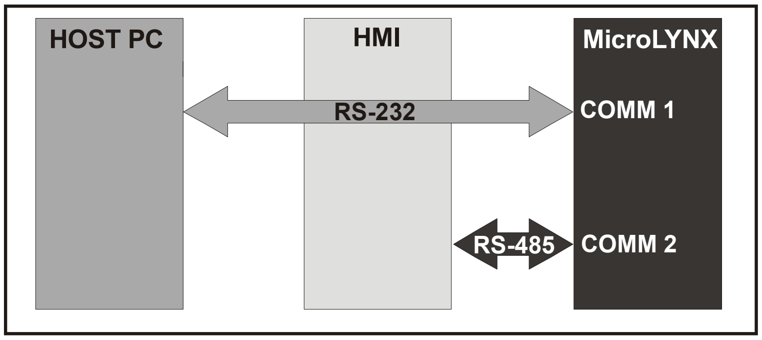


Figure 2.2: Single MicroLYNX/HMI Communications Block Diagram

straight-through connection to the corresponding pins on the P3 connector of the HMI. As is shown in the block diagram illustrated in Figure 2.2, the Host PC communicates through the HMI to the MicroLYNX via RS-232 (MicroLYNX COMM1). HMI specific commands are then forwarded to the HMI via RS-485 (MicroLYNX COMM2).

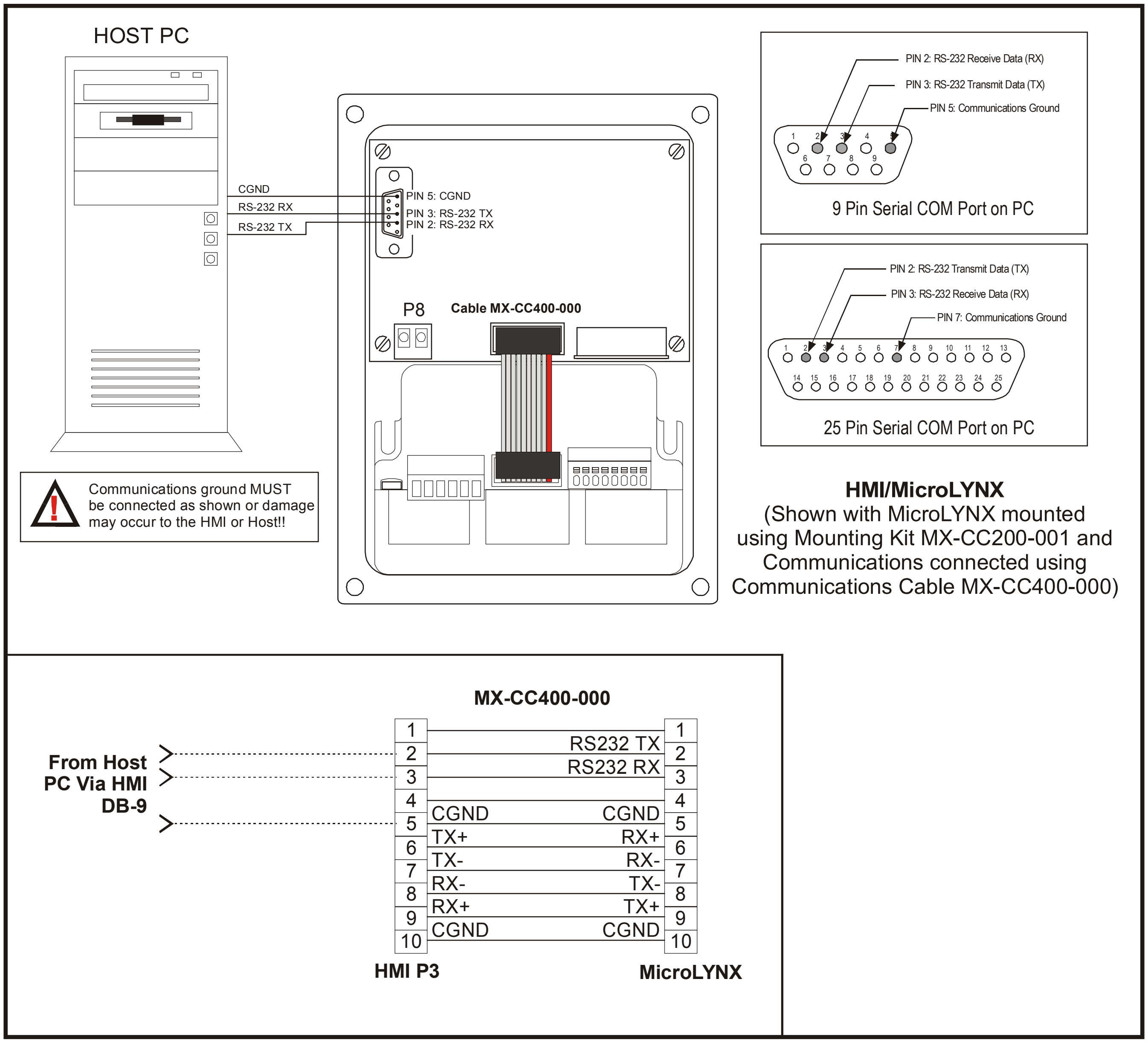

HOST PC
CGND
RS-232 RX
RS-232 TX
PIN 5: CGND
PIN 3: RS-232 TX
PIN 2: RS-232 RX
P8
Cable MX-CC400-000
Communications ground MUST be connected as shown or damage may occur to the HMI or Host!!
PIN 2: RS-232 Receive Data (RX)
PIN 3: RS-232 Transmit Data (TX)
PIN 5: Communications Ground
9 Pin Serial COM Port on PC
PIN 2: RS-232 Transmit Data (TX)
PIN 3: RS-232 Receive Data (RX)
PIN 7: Communications Ground
25 Pin Serial COM Port on PC
HMI/MicroLYNX
(Shown with MicroLYNX mounted using Mounting Kit MX-CC200-001 and Communications connected using Communications Cable MX-CC400-000)
MX-CC400-000
From Host PC Via HMI DB-9
RS232 TX
RS232 RX
CGND
TX+
TX-
RX-
RX+
CGND
RX+
RX-
TX-
TX+
CGND
HMI P3
MicroLYNX

Figure 2.3: Communications Connection, Single MicroLYNX

Connecting Communications, Multiple MicroLYNX Systems

This method illustrates how multiple MicroLYNX Systems may be connected to an HMI using the RS-485 interface.

The illustrations show connection using the optional communication cable MX-CC300-000 for two MicroLYNX systems. If a system contains more than two MicroLYNX systems, the cable may be manufactured by the user to accommodate the system design and configuration. The parts and suppliers of the required cable components are provided in Figure 2.6.

As with the HMI/Single MicroLYNX setup the user's host PC communicates via RS-232 to the host MicroLYNX, which in turn forwards commands to the HMI and additional MicroLYNX system nodes via RS-485.

Figures 2.4 and 2.5 illustrate the connection communications connection for two MicroLYNX Systems.

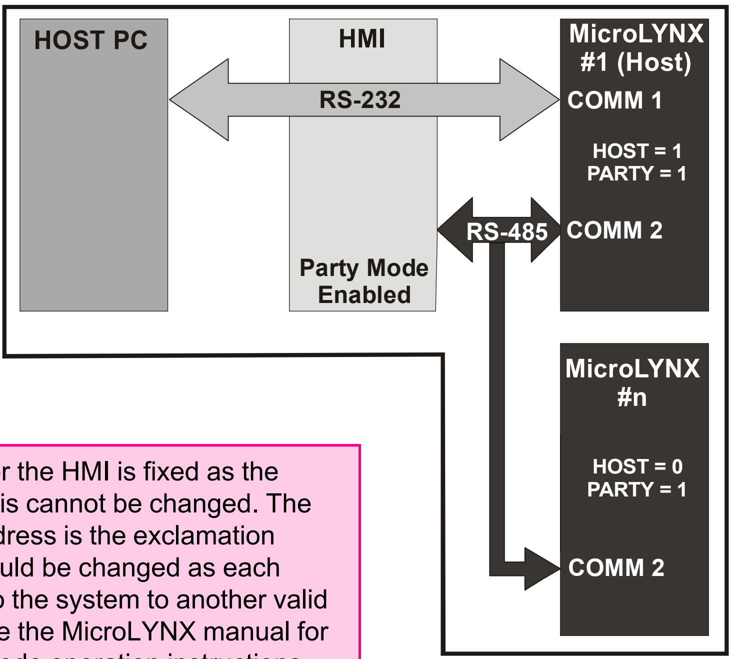


Figure 2.4: Multiple MicroLYNX/HMI Communications Block Diagram

NOTE: The node address for the HMI is fixed as the uppercase character "H". This cannot be changed. The default MicroLYNX node address is the exclamation mark character "!". This should be changed as each MicroLYNX node is added to the system to another valid character other than "H". See the MicroLYNX manual for valid addresses and party mode operation instructions.

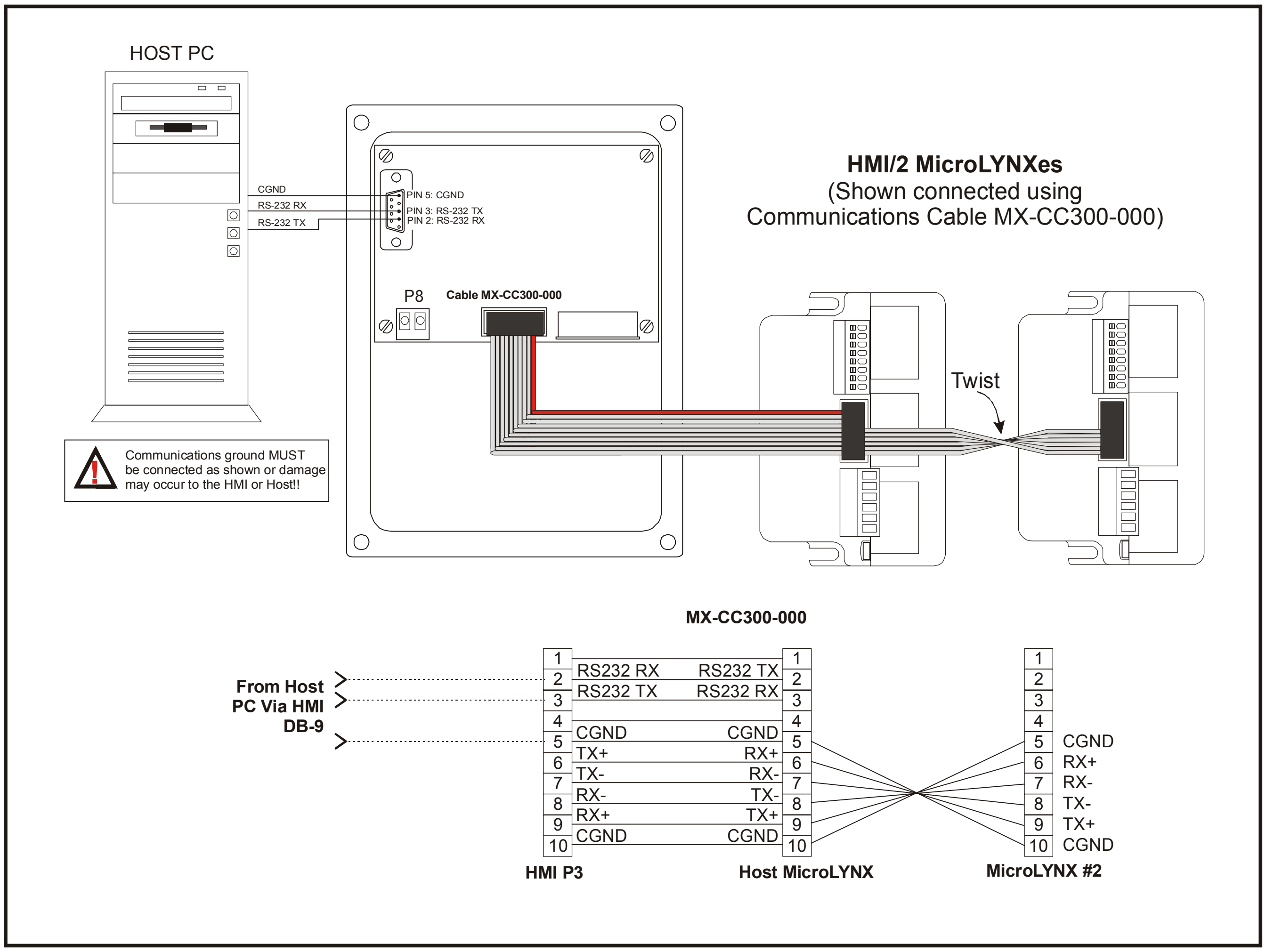


Figure 2.5: Communications Connection, Two MicroLYNX Systems using MX-CC300-000

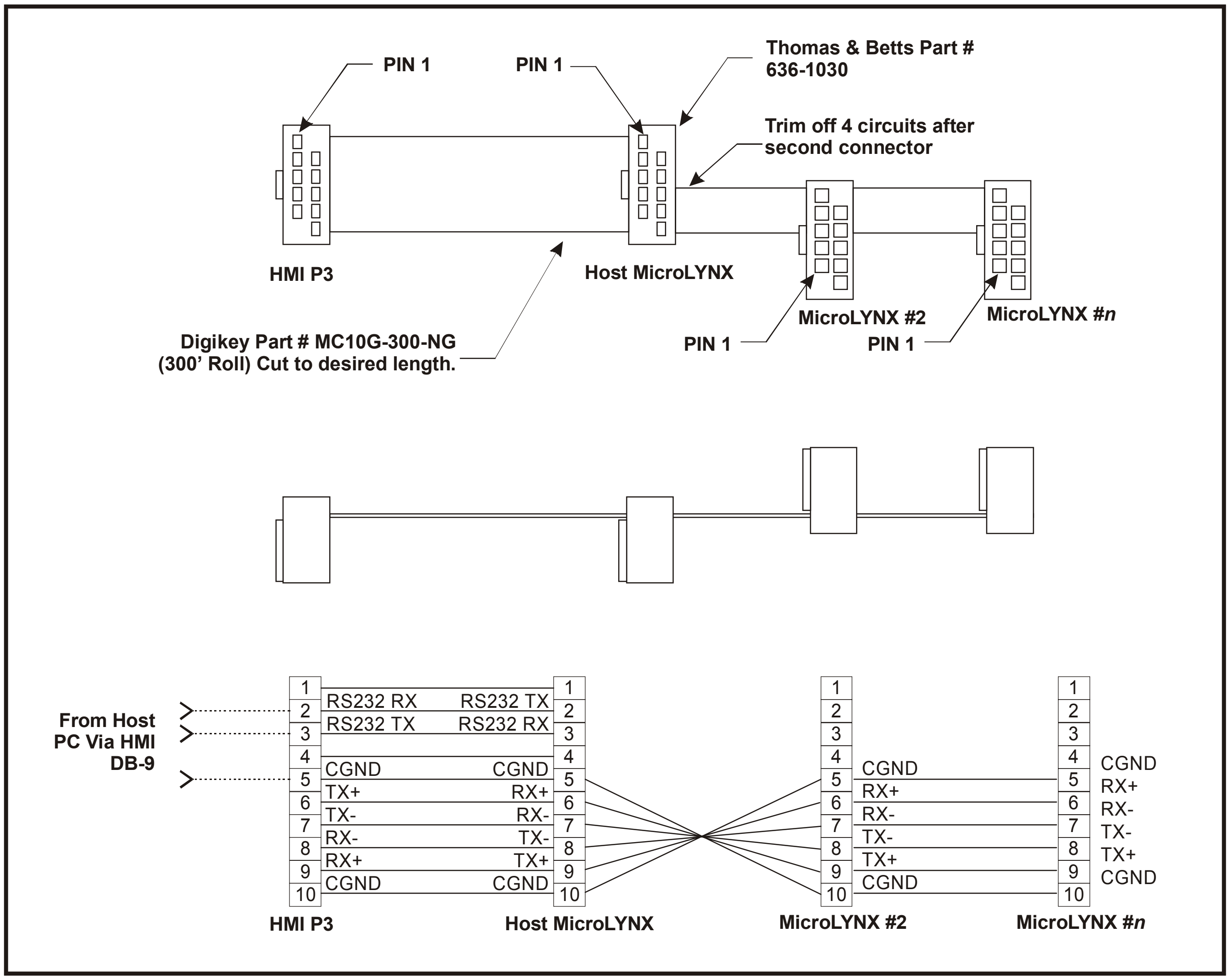
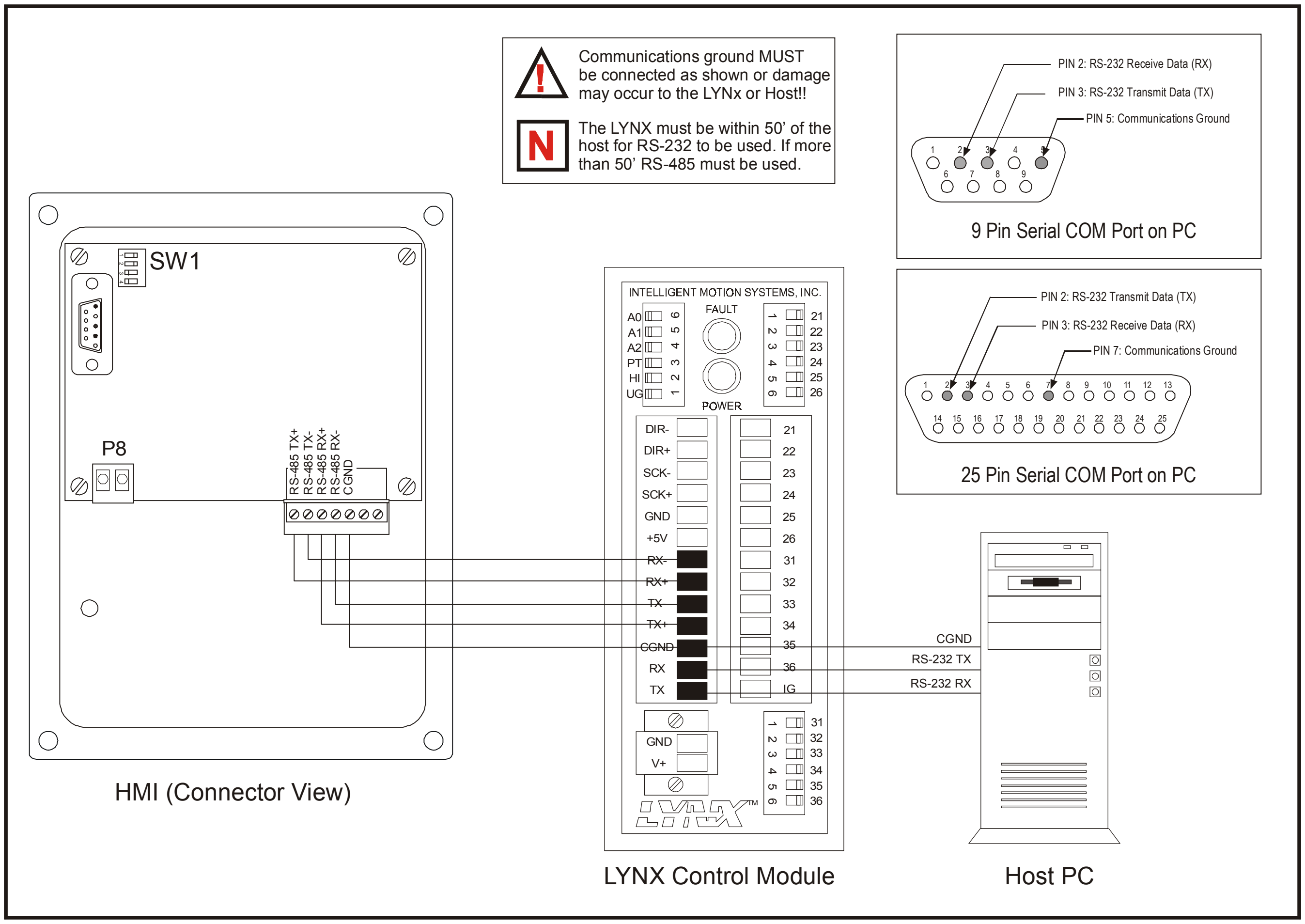


Figure 2.6: Cable Construction for Multiple MicroLYNX/HMI System

RS-485

Figure 2.8 connection diagram illustrates the RS-485 communications interface connected to the HMI using a LYNX Control Module. The HOST PC is connected to the LYNX via COMM 1 RS-232. HMI commands are forwarded to the HMI via LYNX COMM 2, RS-485.



Communications ground MUST be connected as shown or damage may occur to the LYNx or Host!!
The LYNX must be within 50' of the host for RS-232 to be used. If more than 50' RS-485 must be used.
PIN 2: RS-232 Receive Data (RX)
PIN 3: RS-232 Transmit Data (TX)
PIN 5: Communications Ground
9 Pin Serial COM Port on PC
PIN 2: RS-232 Transmit Data (TX)
PIN 3: RS-232 Receive Data (RX)
PIN 7: Communications Ground
25 Pin Serial COM Port on PC
SW1
P8
RS-485 TX+
RS-485 TX-
RS-485 RX+
RS-485 RX-
CGND
INTELLIGENT MOTION SYSTEMS, INC.
FAULT
POWER
DIR-
DIR+
SCK-
SCK+
GND
+5V
RX-
RX+
TX-
TX+
CGND
RX
TX
IG
V+
CGND
RS-232 TX
RS-232 RX
HMI (Connector View)
LYNX Control Module
Host PC

Figure 2.7: Connecting Communications to a Modular LYNX

Connecting the User Output

The HMI features a user output that can be used as an alarm output to turn on a light or activate a siren. This output could also be tied to a LYNX or PLC input or to control system functions. If so used, ensure that the output voltage of the HMI user output does not exceed the maximum rated input voltage of the device being controlled. The output is located at P1, Pin 6.

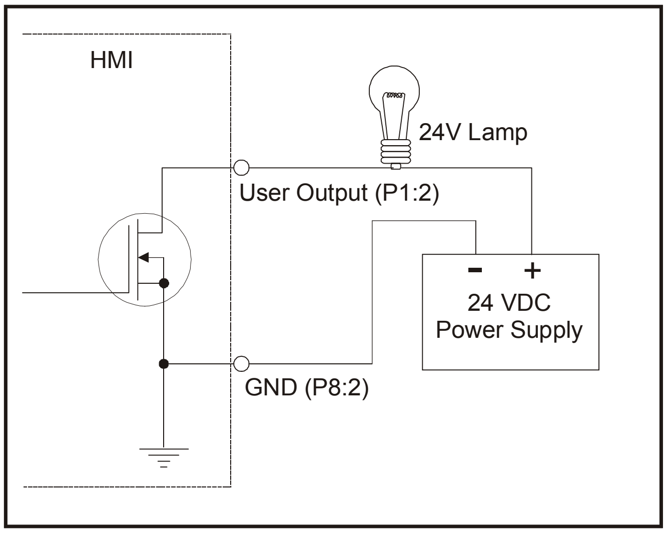


Figure 2.8: User Output Connection

WARNING! When using the user output of the HMI to control a LYNX controller product input, ensure that the voltage applied to the input does not exceed 24VDC!

Mounting the HMI

The HMI is designed to be mounted to a panel or enclosure using standard #6 (M3) hardware. Figure 2.9 below illustrates the cut-out and mounting hole placement dimensions.

A MicroLYNX may also be mounted directly to the HMI to save space. This uses the optional mounting kit MX-CC200-001. The kit consists of a spacer plate and two round slotted nuts. Figure 2.11 illustrates the use of this kit.

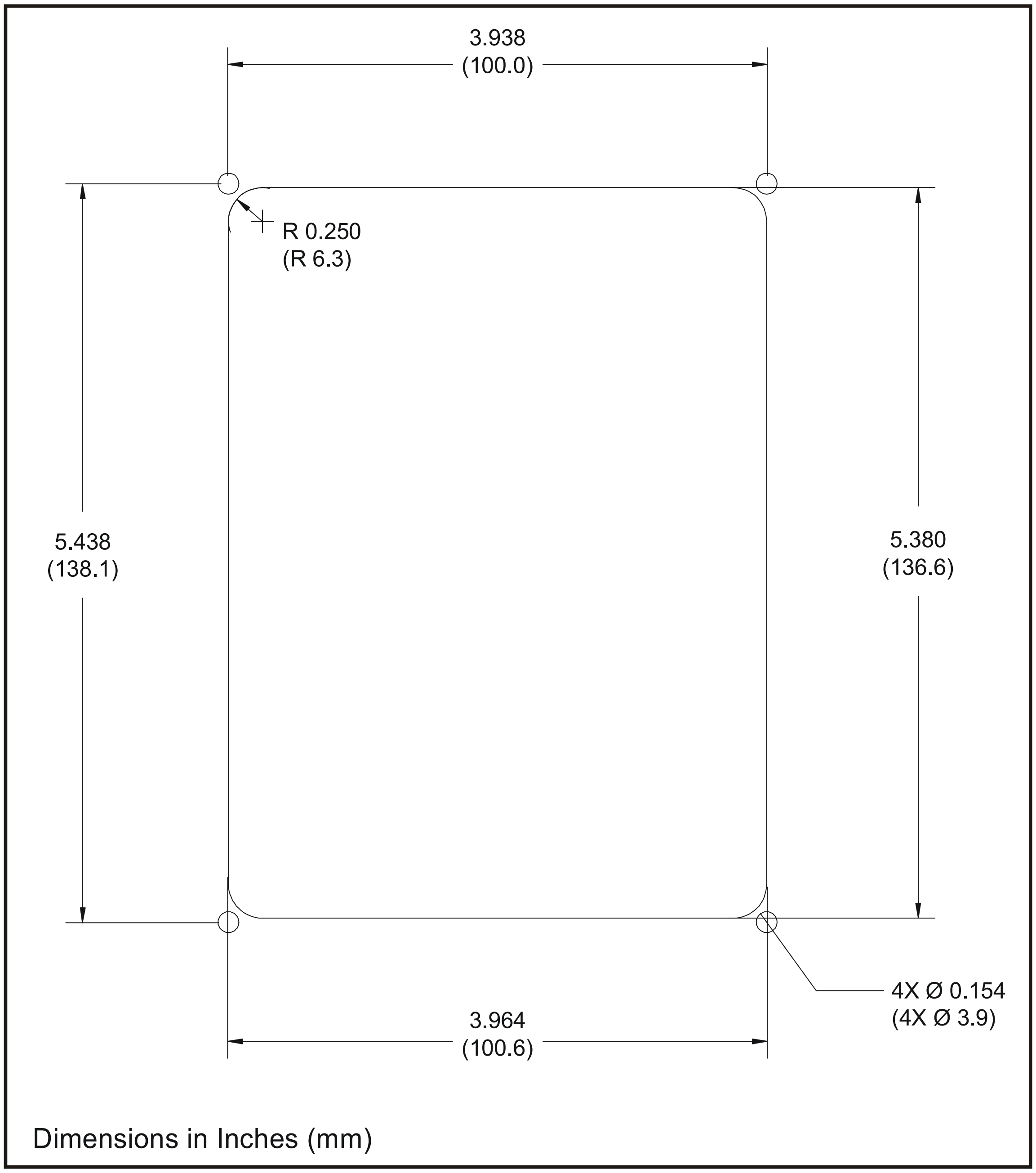


Figure 2.9: HMI Panel Cut-out and Mounting Hole Dimensions

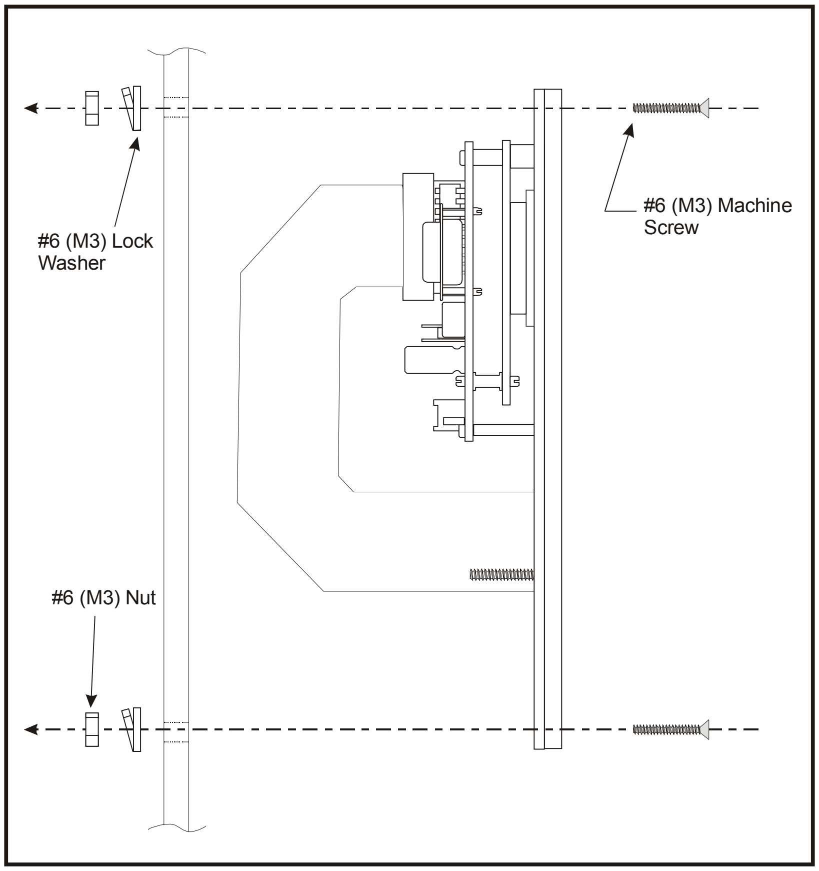


Figure 2.10: HMI Panel Mounting

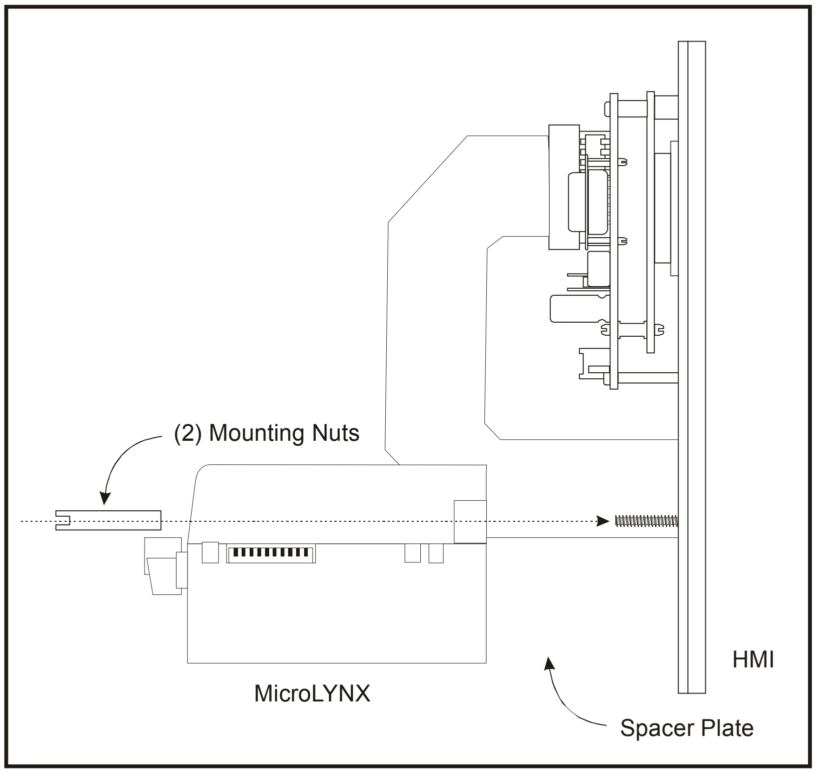


Figure 2.11: Mounting a MicroLYNX to an HMI with the Mounting Hardware Kit MX-CC200-001

SECTION 3

Setup and Operation

Section Overview

This section covers the general setup and operation of the HMI, as well as its three possible modes of operation. Covered are:

- Power-up Defaults.
- Power-up Sequence.
- Establishing Communications.
- Setting Up the HMI.
- Modes of Operation.

Power-up Defaults

The HMI will power-up in the following default condition. Once changed, these settings can be restored to the factory default condition by pressing and holding the following keys while power is coming up: “2”, “3” and “ENTER”.

HMI Power-up Defaults	
Function	**Condition**
Cursor	ON
Blinking Cursor	Enabled
Display	ON
User Output	OFF
Hold Data	Disabled
Contrast*	Level 1
Backlight*	OFF
Carriage Control*	Do Not Respond to CR/LF
BAUD Rate*	9600
Echo*	Local
Prompt Character*	Display Prompt
Master Address*	Not Set
Party Mode*	Disabled
Mode*	Programmable Display
Run Start-up Program*	Disabled
Function Keys (F1 - F4)*	Disabled
Alternate Function Keys (F5 & F6)*	Disabled
***Settings stored in non-volatile memory.**	

Table 3.1: HMI Power-up Default Settings

HMI Power-up Sequence

The HMI powers up in the following sequence:

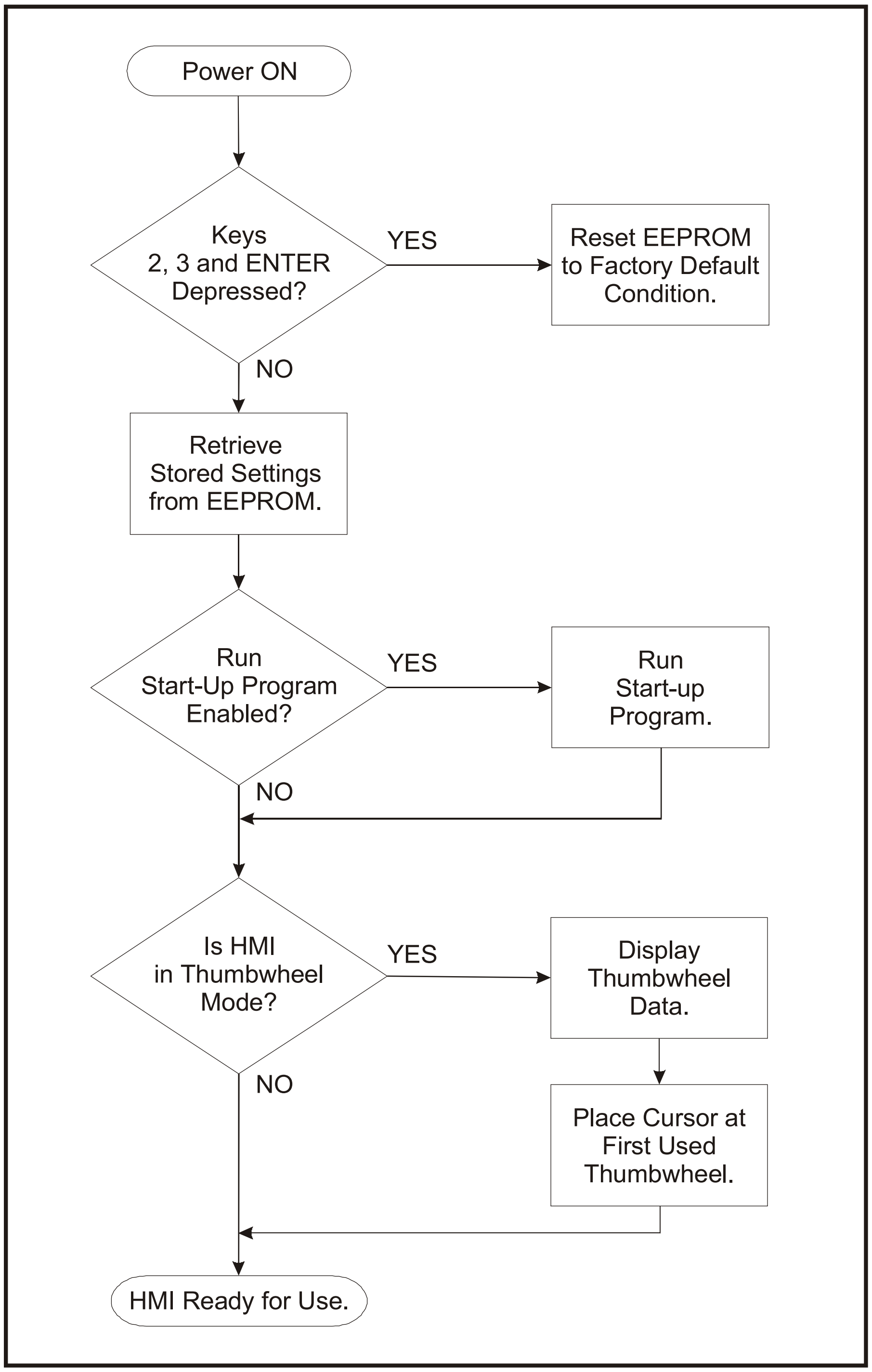

Power ON
Keys 2, 3 and ENTER Depressed?
YES
Reset EEPROM to Factory Default Condition.
NO
Retrieve Stored Settings from EEPROM.
Run Start-Up Program Enabled?
YES
Run Start-up Program.
NO
Is HMI in Thumbwheel Mode?
YES
Display Thumbwheel Data.
Place Cursor at First Used Thumbwheel.
NO
HMI Ready for Use.

Figure 3.1: HMI Power-up Sequence

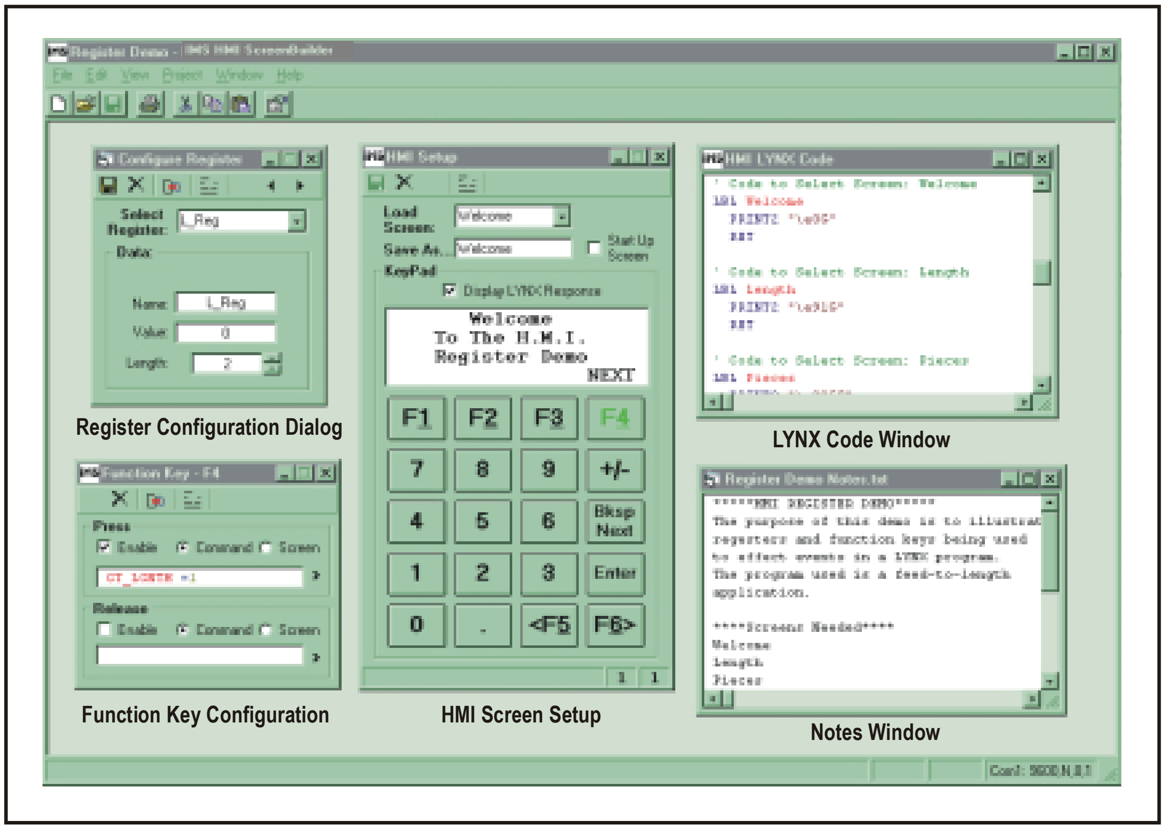
NOTE: To reset the EEPROM to the factory defaults, press and hold “2”, “3” and “ENTER” while power is being applied.

Establishing Communications

Communications can be established with the HMI using either the provided user interface (recommended) or by using a standard ANSI terminal or terminal emulator program in the case that an operating system other than Windows 9x, NT 4.0 or 2000 is used.

The HMI ScreenBuilder

The HMI ScreenBuilder is provided on the LYNX Product Family CD and is the recommended setup and programming interface for the HMI. Its fully graphical interface greatly simplifies the use and programming of the HMI as there is no need to use the escape codes in program development. The various HMI functions are configured using a series of dialogs which allow the user to select from the desired functions and enter the data



```
Welcome
To The H.M.I.
Register Demo
NEXT
```

```
CT_LGNTH =1
```

```
' Code to Select Screen: Welcome
LBL Welcome
  PRINT2 "\e0G"
  RET

' Code to Select Screen: Length
LBL Length
  PRINT2 "\e01G"
  RET

' Code to Select Screen: Pieces
LBL Pieces
```

```
*****HMI REGISTER DEMO*****
The purpose of this demo is to illustrat
registers and function keys being used
to affect events in a LYNX program.
The program used is a feed-to-length
application.

****Screens Needed****
Welcome
Length
Pieces
```

Figure 3.2: The HMI ScreenBuilder

required. The LYNX program code is generated in the the background and displayed in the LYNX code window. The LYNX program code may be exported to a LYNXTerminal file.

Microsoft Windows 9x, NT 4.0 or Windows 2000 is required to use the interface. Installation and use of this utility is covered in depth in *Section 4: The HMI ScreenBuilder*, of this document.

Standard ANSI Terminal Interface

The HMI can be setup and programmed via a standard ANSI Terminal or terminal emulator such as the freely distributed HyperTerminal that is included in the various Microsoft Windows operating systems. Programming the HMI using the terminal interface requires a familiarity with escape codes. The escape codes, as well as several program examples may be found in *Appendix A: Escape Codes.*

PC Direct to HMI

To communicate directly with the HMI, the terminal must be configured as follows:

The escape code entry when connected directly to the HMI is accomplished by pressing the "CTRL" key and the left bracket key " [" simultaneously, releasing the keys and entering the code. For example, to turn on the backlighting for the HMI display you would enter the following:

```
Ctrl+[1l (lowercase letter "L", number 1)
```

No sign-on message will appear in the terminal window when communications is established. Successful connection will be indicated by typed characters appearing in the display on the HMI.

Communication via LYNX Controller Product

PC Direct to HMI Communication Settings	
Connection	Direct to COM<1 - 4>
COM Port Settings	
Bits per second	9600
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None
Terminal Settings	
Emulation	ANSI
NOTE: ASCII Properties need to be set to "Echo Typed Characters Locally", otherwise characters entered will only appear on the HMI screen.	

Table 3.2: Direct Connection PC to HMI Communications Setup

Communication via the LYNX controller product will be the more typical method of communicating to the HMI. This allows the user to program the LYNX controller product as well.

Using this method of communicating with the HMI, commands are sent to the LYNX controller product in either immediate modeor in a program, and then forwarded to the HMI via the PRINT instruction. The INPUT instruction is then used to request data from the HMI, either from a keypad entry or a value stored in an HMI register.

Communication Via LYNX Settings	
Connection	Direct to COM<1 - 4>
COM Port Settings	
Bits per second	9600
Data Bits	8
Parity	None
Stop Bits	1
Flow Control	None
Terminal Settings	
Emulation	ANSI
NOTE: Do NOT echo typed characters to the screen. It will work, but will sure look funny as every character entered will appear twice.	

Table 3.3: Communications via LYNX Settings

Entry of the escape codes is different in this case. Using the same example as before, turning the backlight on would be done like this:

```
PRINT "\e1l"
```

This would turn the backlight on.

Resetting the HMI

The HMI is reset to its factory default state by pressing and holding keys 2, 3 and ENTER while applying power.

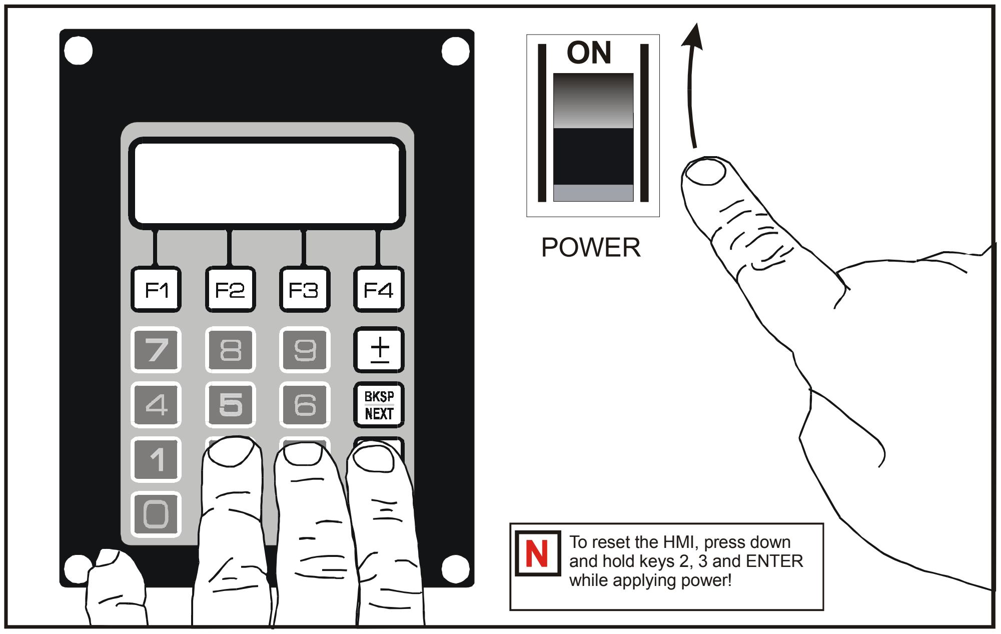

ON
POWER
F1
F2
F3
F4
7
8
9
±
4
5
6
BKSP NEXT
1
0
N
To reset the HMI, press down and hold keys 2, 3 and ENTER while applying power!

Figure 3.3: Resetting the HMI to the Factory Default

User Storage

The user storage consists of 4 kbytes of EEPROM. This storage area is used to store the following information:

- Text and escape codes.
- Text associated with a function key.
- Thumbwheel text labels.
- LYNX Controller Product commands.
- LYNX Controller Product labels.
- User programs.

User programs are defined as any combination of the above items. User programs may not be nested.

The storage area is accessed by address. User storage starts at address 0 and ends at address 4095. The startup program starts at address 0.

Accessing User Storage

User storage is accessed by address. In order to store a program, observe the following:

- Issue the command "Store Program." The first parameter is the program addresses (0 - 4095). The second parameter indicates the program destination; T will indicate "transmit the program results" and L will direct the results locally to the HMI. The last parameter is the program itself. The program may contain ESC codes, register data, and text. The program is terminated with a \t. The register value is embedded with the following command (\exR where x equals the register number (0 - 63)). The reference to the register will expand to the current register value. It is the programmer's responsibility to allow enough space between programs, preventing text from running into another program.

```
Example: PRINT "\e0PT, MOVR 1000\t"
```

Programs in user storage are run with the following command:

- Issue the command Run Program. The only parameter is the program address. The \t is used to terminate the program in user storage and is not transmitted or displayed.

```
Example: PRINT "\e0G"
```

Modes of Operation

The HMI has three modes of operation. These modes can be switched on-the-fly as needed in the user program.

The modes are:

- Programmable Display.
- Thumbwheel Emulation.
- Register.

Programmable Display

When in programmable display, or terminal mode, the HMI will display data received from the serial port at the current cursor position. The cursor will wrap the text from bottom left to top right. The HMI will respond to all escape codes issued when applicable. In programmable display mode, the arrow keys will only operate as alternate function keys.

Programmable display mode is the default mode setting of the HMI. If switching to this mode from another mode the escape code is [ESC]10M, or expressed in LYNX code PRINT "\e10M". This usage would switch the HMI to programmable display mode and not store the setting to NVM.

Thumbwheel Emulation

When placed in thumbwheel emulation mode, the HMI will behave as an electronic thumbwheel. Up to four thumbwheel switch banks, each with up to 10 digits, may be saved. As a thumbwheel is changed the value is written to the EEPROM, thus retaining the value when power is cycled.

Pressing the "NEXT" key will move the cursor from thumbwheel to thumbwheel. The arrow keys will move the cursor within the current thumbwheel. The function keys (F1-F4) may also be programmed and used in thumbwheel emulation mode, however, the alternate function keys (F5-F6) are not available as they are an alternate function of the arrow keys.

Thumbwheel Register Assignment	
Thumbwheel	**Register**
1	0
2	1
3	2
4	3

Table 3.4: Thumbweel Register Assignments

When thumbwheel mode is in use, the HMI automatically switches to local echo. If the HMI mode will be changed on-the-fly, then the desired echo mode will need to be set in the LYNX controller product program. Thumbweel emulation mode establishes registers 0-3 as the registers where the thumbwheel data is stored. Because of this, registers 0-3 should not be modified when switching modes on the fly. See table 3.4 for thumbwheel register assignments.

Observe the following steps to set up thumbwheels. Note that all of the examples are expressed in the form in which they would be entered into LYNX immediate mode or contained within a LYNX controller product program using the "PRINT" instruction.

NOTE: In order to display text when in thumbwheel mode, change to programmable display mode using the "Mode" command. After issuing the desired text display commands, switch back into thumbwheel mode. Ensure that the LYNX program does not modify the value of registers 0 through 3 as these registers are used to hold the thumbwheel values and screen position.

1] Issue the command to clear all thumbwheels used (\e0t). This will clear all thumbwheel values and indicate that no thumbwheels are in use. Example PRINT "\e0t".

2] Issue the command "Store Program at Address 0". This will store the text that is to be displayed for the thumbwheel label.

3] Issue the command "Set Register Value" (\e<0-3>v<val>\b). The first parameter of this command is the thumbwheel register to be used (0-3). The second parameter is the value of the thumbwheel including placeholders. A five digit thumbwheel initialized to -150 would be expressed as -00150. The thumbwheel may contain a maximum of 10 digits plus the sign. If the sign is included with the initial value, the ± key will toggle the sign. If a sign is not included in the initial value, then the ± key will be disabled. For example: PRINT "\e0v5000\b" would set register 0 to 5000.

4] Issue the command "Set Register Screen Position" (\e<0-3>s<row>,<col>\b). This first parameter specifies the thumbwheel register to be used (0-3). The second and third parameters are the row and column locations where the thumbwheel data is to be displayed.

Example: PRINT "\e2s1,1\b" would set register 2 to display at row 1, column 1 of the HMI display.

5] Repeat steps 3 and 4 for each thumbwheel to be used. See the subsection titled *"Function Keys"* in this section of the document for function key setup details.

6] If you desire the HMI to power up in Thumbwheel mode, then the "Mode Command" (\e<mode><flag>M) must be used. The first parameter specifies the HMI mode, the second determines whether or not the mode setting will be stored to non-volatile memory. In this case the command would be expressed thus: PRINT "\e01M".

The current value of the thumbwheel may be accessed by issuing the command "Get Thumbwheel Data" (\e<1-4>W). The current value stored in the thumbwheel register will be transmitted to the LYNX controller product. For example: PRINT "\e3W" would send the value of thumbweel #3, or register 2, to the LYNX.

Register Mode

The HMI features 64 registers numbered 0 - 63 that can be used to store numeric data with a maximum digit limit of 10 digits plus the sign. This mode has the following features:

- Maximum of 4 registers can be displayed at one time. The registers being displayed are referred to as active registers.
- The "NEXT" key moves the cursor from active register to active register.

- The LYNX can change the value of an active register and the displayed value is updated on the screen.
- Displayed register values can be changed via the keypad.
- Non active registers may be changed on-the-fly.
- Programs can be attached to a register. Once the cursor is on the register and a defined function key is pressed, a program associated with this register can be run. The register program address works as follows:

  Mode 1: Address = 4097 .... Store the register value, do not transmit it.

  ```
  Example: PRINT "\e5a4097\b" (Store the value of register 5, don't
  transmit)
  ```

  Mode 2: Address = 0 – 4095 .... Run program at specified address.

  ```
  Example: PRINT "\e5a2000\b" (Run the program associated with register 5
  at address 2000)
  ```

  Mode 3: Address = 4096 .... Store register value, transmit it.

  ```
  Example: PRINT "\e5a4096\b" (Store the value of register 5 and transmit
  it)
  ```

- Register mode switches to local echo. If the mode is changed on-the-fly, the LYNX program needs to set the desired echo mode.

In order to setup the HMI in Register Mode, the following steps need to be followed for each used register:

1. Issue the command Set Register Value.

   PRINT "\e5v150000\b"   'Set register 5 to 150000.
2. Issue the command Set Register Screen Position.

   PRINT "\e5s1,12\b"   'Position register 5 at row 1 column 12.
3. Issue the command Set Register Program Address.

   PRINT "\e5a2000\b"   'Set register to program address 2000.
4. Issue the command Store Program if the register is to run a program.

   PRINT "\e200PT,STARTLX\t"   'Store string "STARTLX" at address 200, transmit.

In order to run in Register Mode, the following steps need to be followed:

1. Issue the command Mode, switch into register mode.

   PRINT "\e21M"   'Set register mode, write mode to NVM.
2. Issue the command Set No Registers Active.

   PRINT "\e0u"   'Set no registers active.
3. Issue the command Set Register Transmit Method. Then enable only the function keys used.

   PRINT "\e0x"   'Set register to transmit when the "Enter" key is pressed.
4. Display the register prompts.
5. Issue the command Set Register Active and Display.

PRINT "e5U"   'Set register 5 active and display.

Once register mode is active, the NEXT key will move the cursor from register to register. In order to change a register value, move the cursor to the desired register and type the new value. The cursor will change shape indicating register edit mode; press ENTER to exit register mode. The cursor will change back to the blinking bar. When the key that was defined using the Set Register Transmit Method is pressed, the program associated with the active register (the register that the cursor is on) is executed.

Note: This mode allows a real time display to be created.

Function Keys

The HMI has four fixed and two alternate function keys. This allows 6 press and/or release functions per screen. These functions may be changed on-the-fly in response to user actions. The function keys are programmed via pointers to user storage. Each function key may also be individually enabled or disabled. If using the recommended programming interface, the HMI ScreenBuilder, there is a dialog which eases the configuration of each function keys from screen to screen.

In order to program the function keys, observe the following steps:

1. Issue the command Store Program. The program may contain ESC codes, register data, and text. The data from the HMI may run locally or transmitted to the LYNX.

   PRINT "e\2000PT,RUNPROG\t" 'Store string "RUNPROG" at address 2000, transmit.

   ("RUNPROG" is representative of a LYNX program label)

2. Issue the command Set Function Key Program Address Pointer. This command will set the program address to be run when the function key is pressed or released (address programmed in step 1).

   PRINT "e\1fP,2000\b" ' Point function key F1 press function to address 2000.

3. Issue the command Function Key Enable/Disable and/or Alternate Function Key Enable/Disable in order to enable the programmed function keys.

   PRINT "e\16F" 'Enable the press function of F1.

This now completes function key setup. Pressing a programmed function key will run the program entered in step 1. The \t is used to terminate the program in user storage and is not transmitted or displayed.

SECTION 4

The HMI ScreenBuilder

Section Overview

The HMI ScreenBuilder is the recommended programming interface for the IMS Human Machine Interface. The HMI ScreenBuilder's key feature is that it eliminates the need of using escape codes for configuring and programming the HMI.

The HMI ScreenBuilder will generate all of the necessary code for the HMI to operate in concert with a LYNX controller product. It will also generate the LYNX code needed to call up the various screens in the project in the form of subroutines that can exported to a LYNX terminal file.

The HMI ScreenBuilder will not create the LYNX program itself. This must be created by the user outside of the HMI ScreenBuilder program. The HMI ScreenBuilder simplifies this task by allowing the user to export the LYNX code window to either a LYNX Terminal editor file (*.lxt) or ASCII text (*.txt) to be used as a template for the final LYNX program.

Covered in this section are:

- Installing and Configuring the HMI ScreenBuilder.
- The HMI ScreenBuilder Main Window
- The HMI Setup Window.
- The Function Key Setup Window.
- The Configure Register Window.
- The LYNX Code Window.
- The Notes Window.
- Putting It All Together - Building Projects.
- Project Demonstration.

Installing and Configuring the HMI ScreenBuilder

The HMI ScreenBuilder utility is free and may be obtained from either the LYNX CD, which ships with the LYNX controller products, or it may be downloaded from the software page of the IMS web site.

Minimum System Requirements

- IBM Compatible 486 or Higher PC.
- Windows 95/98, Windows NT4.0 Service Pack 3 or higher, or Windows 2000.
- 5 MB hard drive space.
- A free serial communications port.

Installation

To install the HMI ScreenBuilder to your hard drive, insert the CD into your CD-ROM Drive. The 3.5" CD, while smaller than typical compact disks, will work in any horizontally mounted, tray-type CD drive.

To start the installation click "Start > Run" and type "[Drive Letter]:\HMI ScreenBuilder\Setup.exe" in the "Open" box.

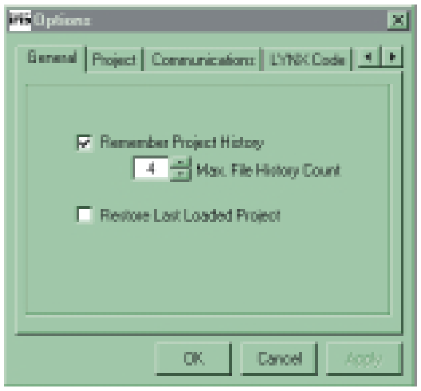
Follow the on-screen instructions to complete the installation.

Preferences Configuration

Once installed, you may start the HMI ScreenBuilder utility by clicking “Start>Programs>HMI ScreenBuilder>HMI ScreenBuilder”.

Initially, the HMI ScreenBuilder will not be communicating with the HMI. Communications will not be initiated until the user downloads the screens to the HMI.

The dialogs shown in figure 4.1 are opened by selecting “Edit>Preferences” on the menu bar. Some of these preferences are project specific and will only be active if a project is open.

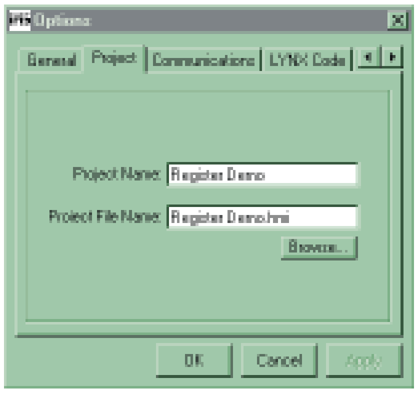


☑ Remember Project History

4 Max. File History Count

☐ Restore Last Loaded Project

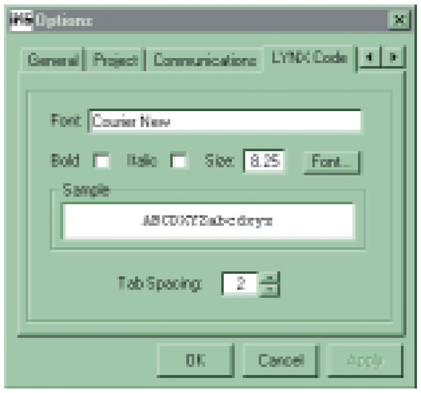
General Properties



Project Name: Register Demo

Project File Name: Register Demo.hmi

Project Properties



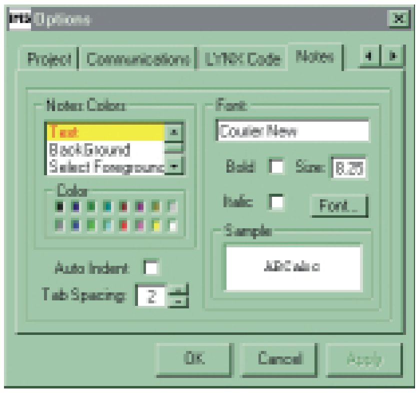
Font: Courier New

Bold ☐ Italic ☐ Size: 8.25

Sample: ABCDXYZabcdxyz

Tab Spacing: 2

LYNX Code Window Options



Font: Courier New

Bold ☐ Size: 8.25

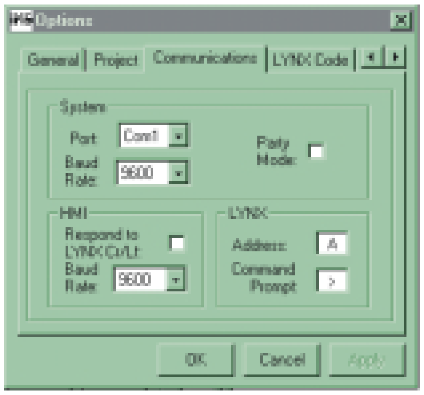
Italic ☐

Sample: ABCabc

Auto Indent ☐

Tab Spacing: 2

Note Window Options



Port: Com1

Baud Rate: 9600

Parity Mode ☐

Respond to LYNX Cr/Lf ☐

Baud Rate: 9600

Address: A

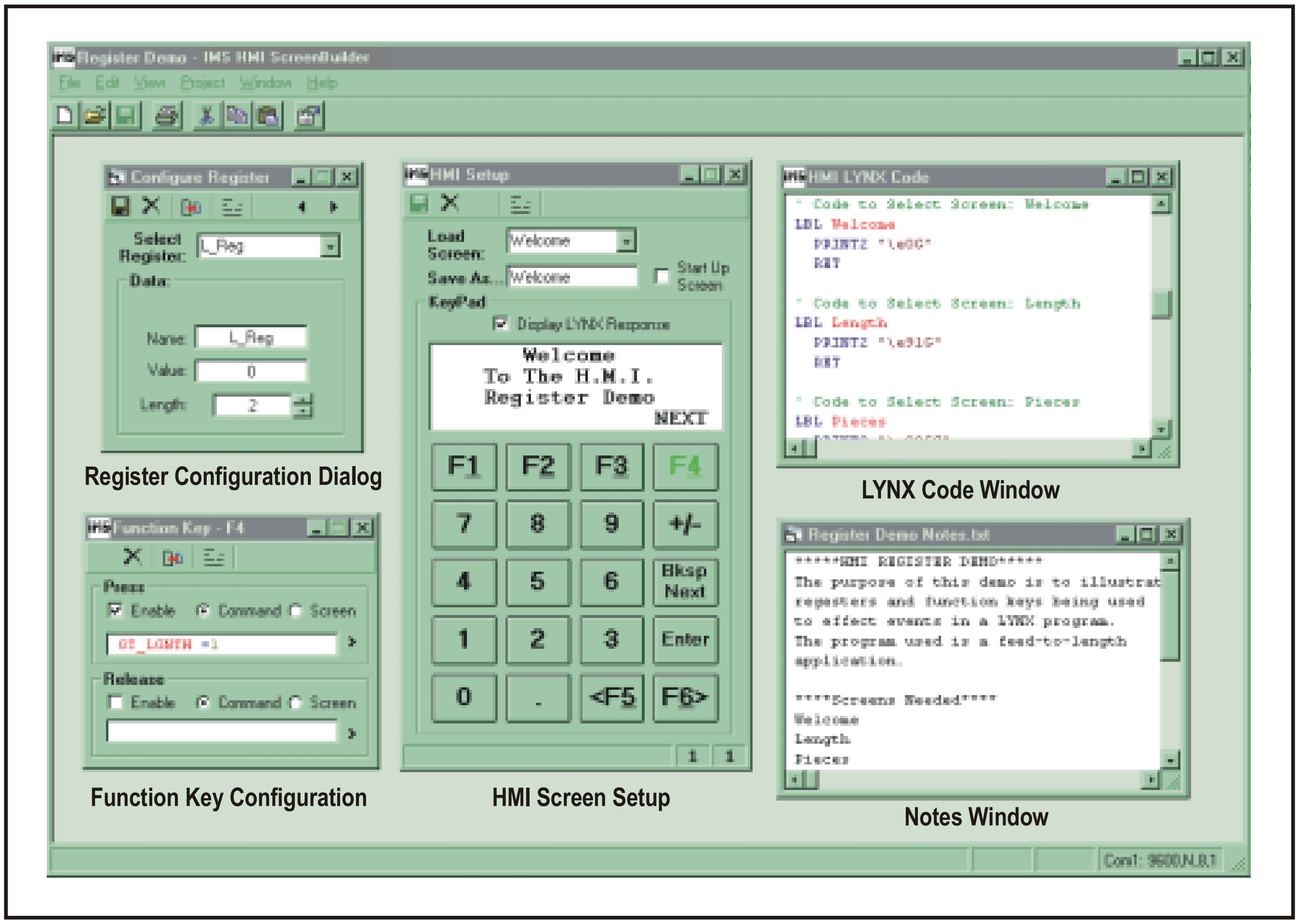
Command Prompt: >

Communications Preferences

Figure 4.1: HMI Project Preferences

HMI ScreenBuilder Main Window

The main window of the HMI ScreenBuilder has several components used to create the HMI Project. These components allow the user to set up the various features of the HMI such as screens, registers and function keys. The components of the main window are each covered in detail in separate subsections. They are listed here in overview:



Select Register: L_Reg
Data:
Name: L_Reg
Value: 0
Length: 2

Press
[✓ Enable] [✓ Command] Screen
GT_LGNTH =1
Release
Enable [✓ Command] Screen

Load Screen: Welcome
Save As...: Welcome
Start Up Screen
KeyPad
[✓ Display LYNX Response]

```
    Welcome
 To The H.M.I.
 Register Demo
             NEXT
```

```
' Code to Select Screen: Welcome
LBL Welcome
  PRINT2 "\e0G"
  RET

' Code to Select Screen: Length
LBL Length
  PRINT2 "\e31G"
  RET

' Code to Select Screen: Pieces
LBL Pieces
```

```
*****HMI REGISTER DEMO*****
The purpose of this demo is to illustrat
registers and function keys being used
to effect events in a LYNX program.
The program used is a feed-to-length
application.

****Screens Needed****
Welcome
Length
Pieces
```

Conf1: 9600,N,8,1

Figure 4.2: HMI ScreenBuilder Main Window

HMI Setup

The HMI Setup dialog is central to the HMI ScreenBuilder project. This dialog will be used to create and edit the various screens which make up the project.

Function Key Setup

There are six function key setup dialogs, each identical in appearance and function. They are activated by either clicking on the function key on the HMI setup window, or by selecting "View>Configure Function Keys>F<1-6>" on the main screen menu bar. Once configured, the function key must be applied to the active screen in order to function.

Configure Register

The configure register dialog configures the HMI registers. Registers may be given a unique name consisting of 1 - 8 alpha-numeric characters following the LYNX labeling rules. Registers must be saved and applied to the active screen. This dialog is opened by either right clicking in the HMI Setup screen at the location where you want the register to be placed, or by selecting “View>Configure Registers”.

Registers may also be deleted form this dialog. When a register is deleted, it will be removed from every screen where it is placed, not only the active screen.

The last column, column 20, is a register dead spot. Register entries made in this column will cause the register value to revert to 0 when entered.

Notes Window

The notes window is a convenient place to take notes concerning the project. The contents of the notes window is automatically saved as “<project filename> Notes.txt” in the project directory. It may be printed by selecting “File>Print>Note Window” on the main window menu bar. The notes window may be viewed by selecting “View>Note Window”.

LYNX Code Window

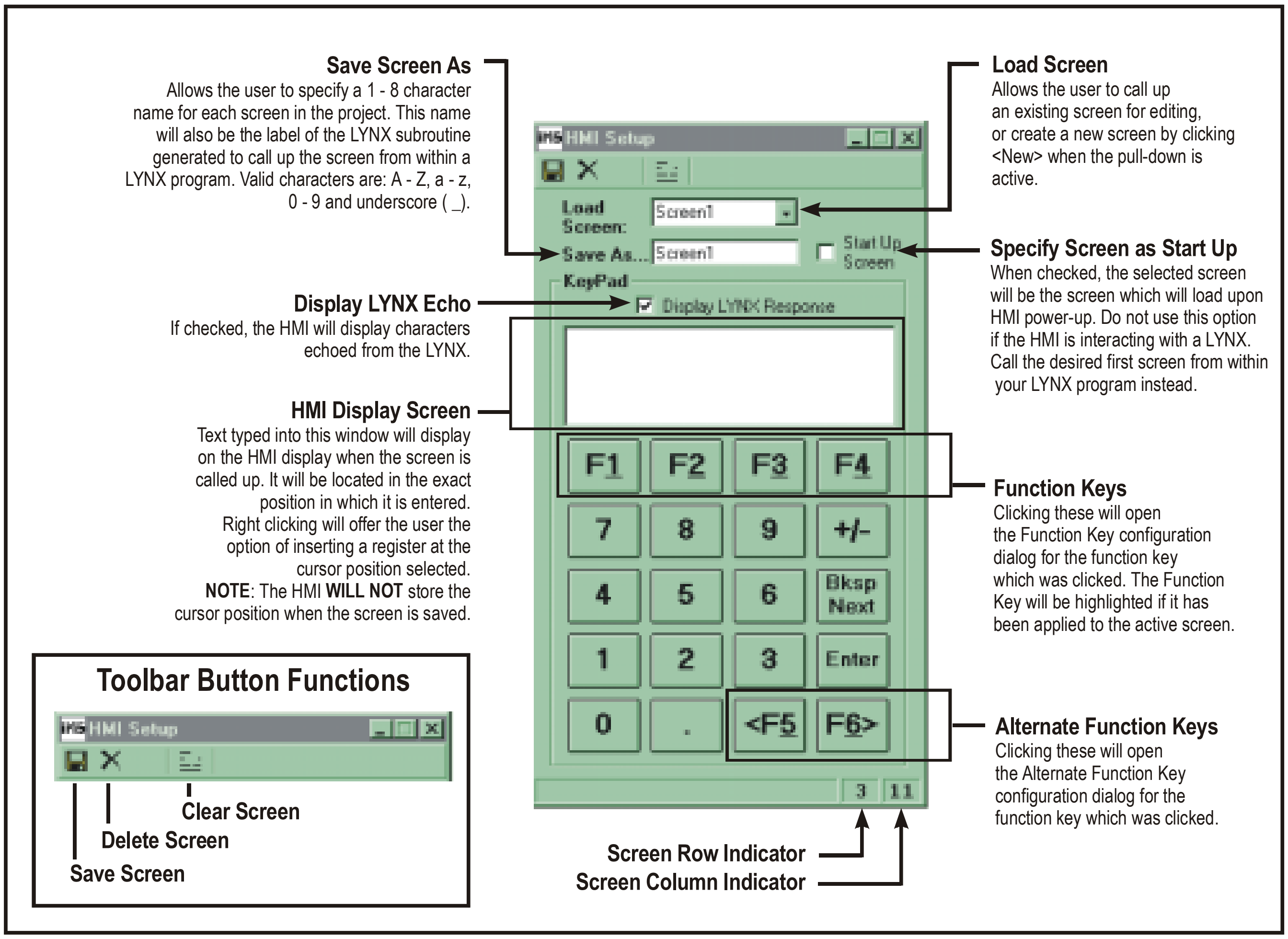
The LYNX code window represents one half of the final output of the HMI ScreenBuilder. This window contains the generated LYNX code which will be the template for your LYNX program. This window can be exported to either an ASCII text (*.txt) file, or a LYNX Terminal Editor (*.lxt) file. The advantage of the LYNX Terminal format is that the color coding is retained. The code is exported by selecting “File>Save LYNX Code As”.

To open the LYNX code window, select “View>LYNX Code Window”. The HMI must be updated or the project compiled before the code can be used in a LYNX program.

The HMI Setup Window

The HMI Setup window is the primary window that will be used in the creation of your HMI ScreenBuilder project. This is the window that will be used to create the various screens that will make up the project. The Function Key setup dialog and the Configure Register dialog are both subordinate to this window.

Precise details about each screen component are located in figure 4.3.



Save Screen As
Allows the user to specify a 1 - 8 character name for each screen in the project. This name will also be the label of the LYNX subroutine generated to call up the screen from within a LYNX program. Valid characters are: A - Z, a - z, 0 - 9 and underscore (_).

Display LYNX Echo
If checked, the HMI will display characters echoed from the LYNX.

HMI Display Screen
Text typed into this window will display on the HMI display when the screen is called up. It will be located in the exact position in which it is entered. Right clicking will offer the user the option of inserting a register at the cursor position selected.
NOTE: The HMI **WILL NOT** store the cursor position when the screen is saved.

Toolbar Button Functions
Clear Screen
Delete Screen
Save Screen

Load Screen
Allows the user to call up an existing screen for editing, or create a new screen by clicking <New> when the pull-down is active.

Specify Screen as Start Up
When checked, the selected screen will be the screen which will load upon HMI power-up. Do not use this option if the HMI is interacting with a LYNX. Call the desired first screen from within your LYNX program instead.

Function Keys
Clicking these will open the Function Key configuration dialog for the function key which was clicked. The Function Key will be highlighted if it has been applied to the active screen.

Alternate Function Keys
Clicking these will open the Alternate Function Key configuration dialog for the function key which was clicked.

Screen Row Indicator
Screen Column Indicator

Figure 4.3: HMI Setup Window

NOTE: Do not specify a screen as a “Start Up” screen if the HMI will be used exclusively to affect events inside a LYNX program. Use the “CALL” instruction in your LYNX program to call up the desired start up screen.

NOTE: Function key and register settings must be applied to the active screen, and the screen saved in order to be used. Saving the screen does not save the project.

The Function Key Setup Window

There are six separate function key setup dialogs, four for the standard function keys (F1 - F4) and two for the alternate function keys (F5 - F6). These dialogs are identical in form and function. Subordinate to the Function Key setup window is the LYNX command worksheet, a tool which allows the user to easily configure the functions of the keys.

Each function key setup window has two main areas:

1. **Press**

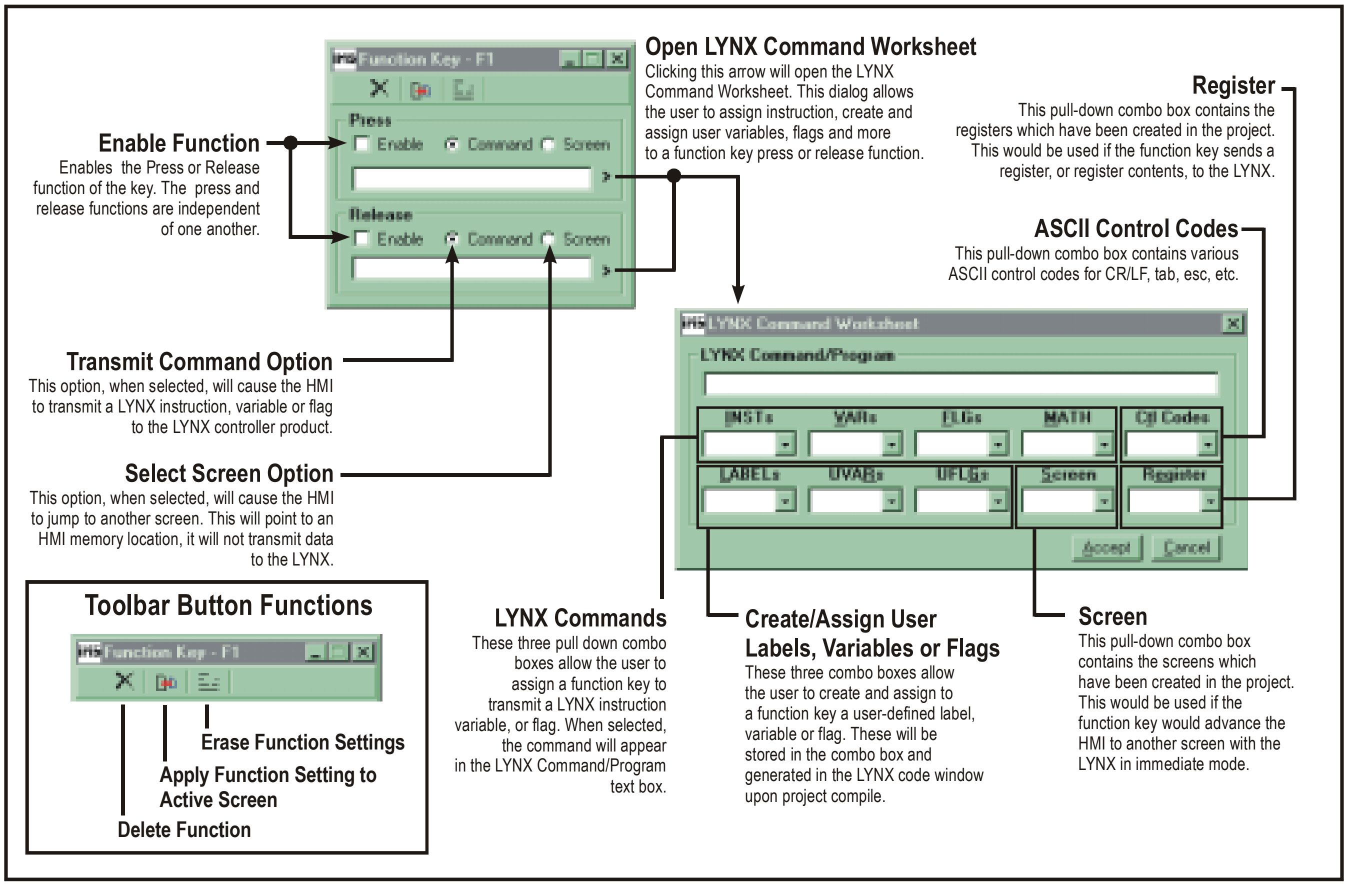
   The press function is the command which will execute upon pressing the function key.

2. **Release**

   The release function is the command which will execute upon the release of the function key.

The press or release function key must be applied to the active screen in order for the fuction to be accessed. If the function is repeated for multiple screens then it must be re-applied to each screen.

Precise details about each screen component are located in figure 4.4.



Open LYNX Command Worksheet
Clicking this arrow will open the LYNX Command Worksheet. This dialog allows the user to assign instruction, create and assign user variables, flags and more to a function key press or release function.

Register
This pull-down combo box contains the registers which have been created in the project. This would be used if the function key sends a register, or register contents, to the LYNX.

Enable Function
Enables the Press or Release function of the key. The press and release functions are independent of one another.

ASCII Control Codes
This pull-down combo box contains various ASCII control codes for CR/LF, tab, esc, etc.

Transmit Command Option
This option, when selected, will cause the HMI to transmit a LYNX instruction, variable or flag to the LYNX controller product.

Select Screen Option
This option, when selected, will cause the HMI to jump to another screen. This will point to an HMI memory location, it will not transmit data to the LYNX.

Toolbar Button Functions
Erase Function Settings
Apply Function Setting to Active Screen
Delete Function

LYNX Commands
These three pull down combo boxes allow the user to assign a function key to transmit a LYNX instruction variable, or flag. When selected, the command will appear in the LYNX Command/Program text box.

Create/Assign User Labels, Variables or Flags
These three combo boxes allow the user to create and assign to a function key a user-defined label, variable or flag. These will be stored in the combo box and generated in the LYNX code window upon project compile.

Screen
This pull-down combo box contains the screens which have been created in the project. This would be used if the function key would advance the HMI to another screen with the LYNX in immediate mode.

Figure 4.4: Function Key Setup Dialog

NOTE: The desired function of the key must be enabled and applied to the active screen for it to operate. If a function key will have the same function throughout the project, it must be applied to each screen separately.

The Configure Register Window

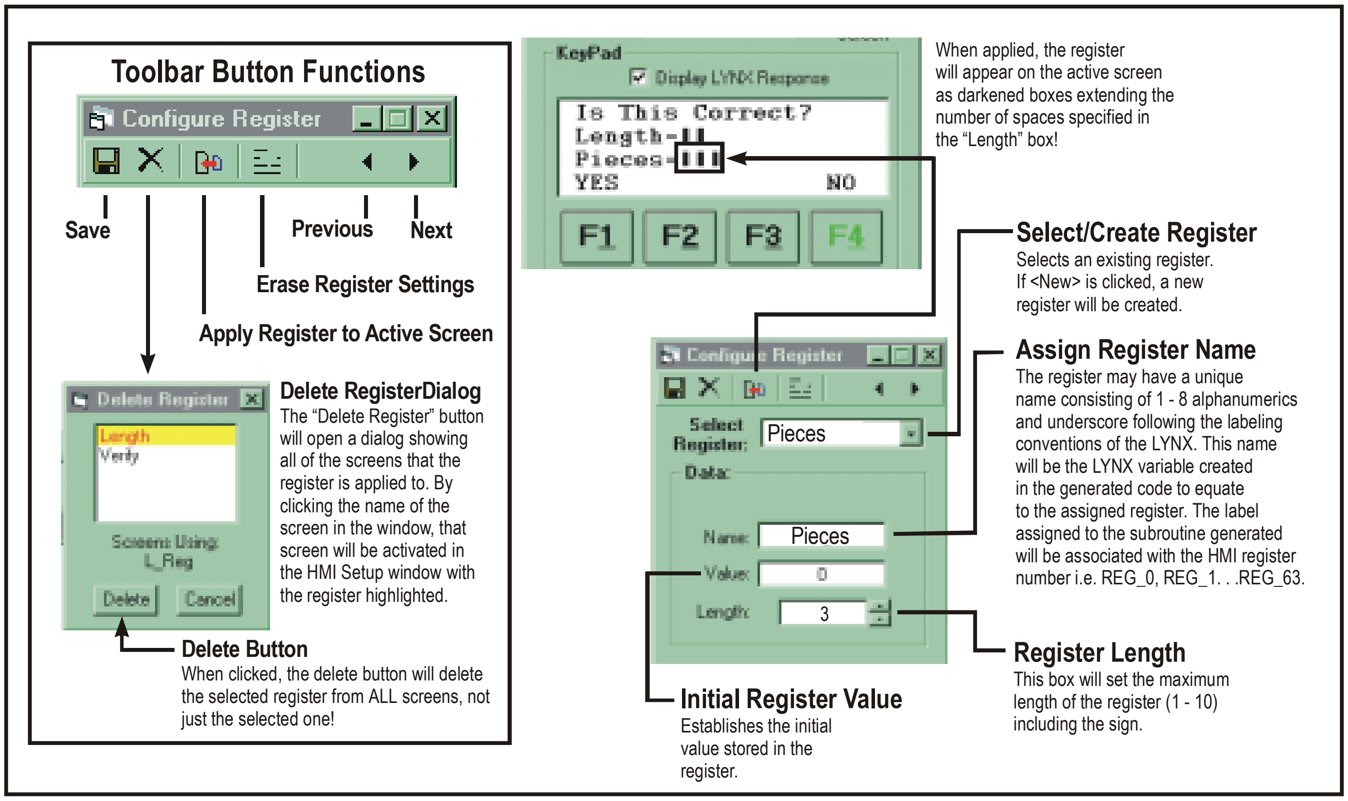
The configure register dialog configures the HMI registers. A register is used to hold numeric data. This data can be received from, or transmited to, a LYNX variable. The user can configure up to 64 registers numbered 0 - 63. Each register can have a maximum of 10 digits not including the number sign. When setting the register length, be sure to leave a space for the sign if needed.

Using the HMI ScreenBuilder, registers may be given a unique name consisting of 1 - 8 alpha-numeric characters following the LYNX labeling rules. This name will be created as a user variable in the LYNX code window. This variable will be attached to the register in the generated subroutine. The subroutine itself will be labeled by the register number i.e. REG_0, REG_1. . . REG_63.

Registers must be saved and applied to the active screen. This dialog is opened by either right clicking in the HMI Setup screen at the location where you want the register to be placed, or by selecting “View>Configure Register”.

Precise details about each register configuration screen component are located in figure 4.5.

USAGE NOTE: Registers may not be given the same name as a LYNX factory instruction, variable or flag.



Delete RegisterDialog
The “Delete Register” button will open a dialog showing all of the screens that the register is applied to. By clicking the name of the screen in the window, that screen will be activated in the HMI Setup window with the register highlighted.

Delete Button
When clicked, the delete button will delete the selected register from ALL screens, not just the selected one!

When applied, the register will appear on the active screen as darkened boxes extending the number of spaces specified in the “Length” box!

Select/Create Register
Selects an existing register. If <New> is clicked, a new register will be created.

Assign Register Name
The register may have a unique name consisting of 1 - 8 alphanumerics and underscore following the labeling conventions of the LYNX. This name will be the LYNX variable created in the generated code to equate to the assigned register. The label assigned to the subroutine generated will be associated with the HMI register number i.e. REG_0, REG_1. . .REG_63.

Register Length
This box will set the maximum length of the register (1 - 10) including the sign.

Initial Register Value
Establishes the initial value stored in the register.

Figure 4.5: Configure Register Dialog

USAGE NOTE: The very last space on the end of each row is a register “dead spot”. No registers may be placed in this space.

The LYNX Code Window

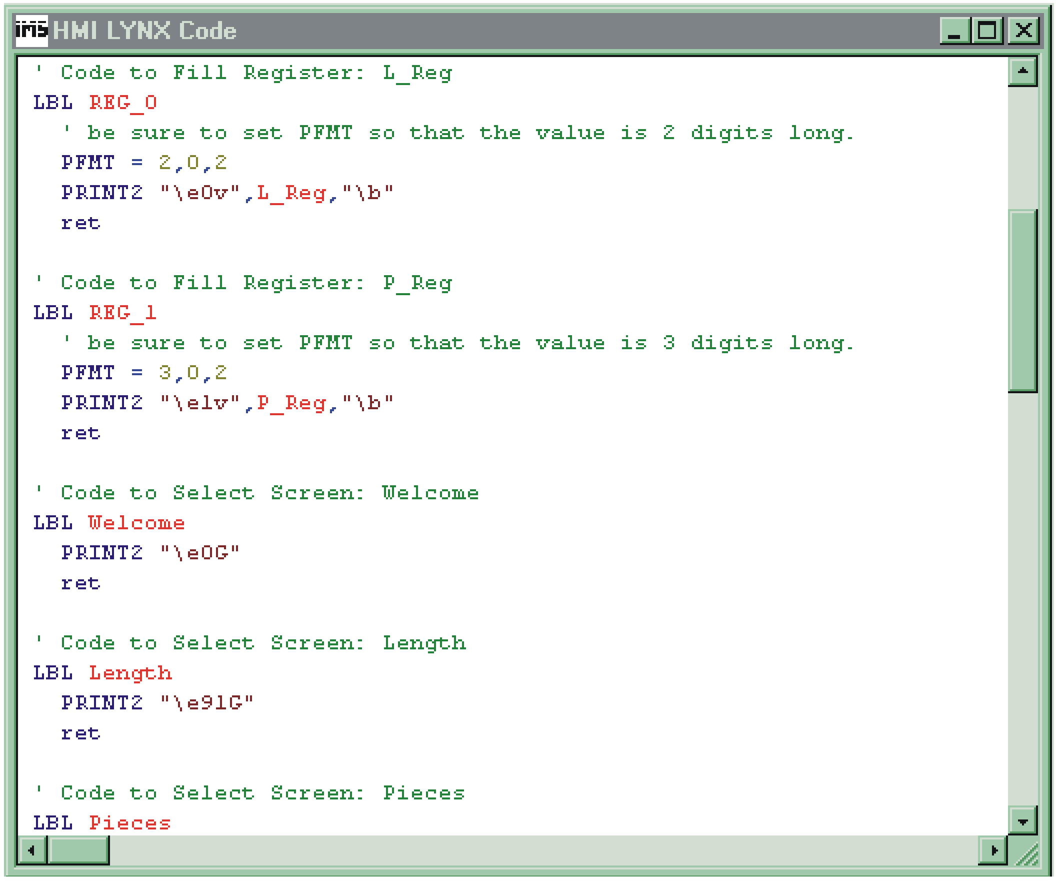
The LYNX code window represents one half of the final output of the HMI ScreenBuilder. This read-only window contains the generated LYNX code which will be the template for your LYNX program.

The code generation occurs either when the project is compiled or the HMI is updated. To compile the project select "Project>Compile". To update the HMI select "Project>Update HMI>Direct or Through LYNX".

This window can be exported to either an ASCII text (*.txt) file, or a LYNX Terminal Editor (*.lxt) file. The advantage of the LYNX Terminal format is that the color coding is retained. The code is exported by selecting "File>Save LYNX Code As".

To open the LYNX code window, select "View>LYNX Code Window". The project must be compiled before the code can be used in a LYNX program.

The LYNX code window can also be printed by selecting "File>Print>LYNX Code Window". The window must be open in order to print the LYNX code.



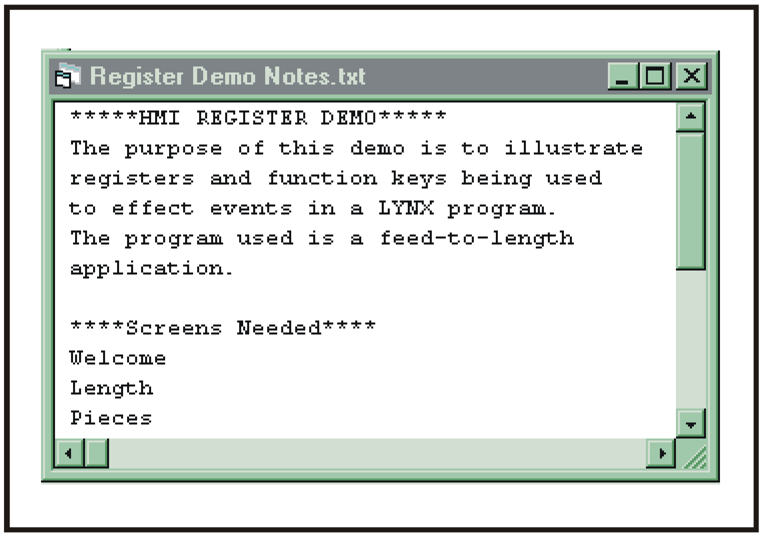
```
' Code to Fill Register: L_Reg
LBL REG_0
  ' be sure to set PFMT so that the value is 2 digits long.
  PFMT = 2,0,2
  PRINT2 "\e0v",L_Reg,"\b"
  ret

' Code to Fill Register: P_Reg
LBL REG_1
  ' be sure to set PFMT so that the value is 3 digits long.
  PFMT = 3,0,2
  PRINT2 "\e1v",P_Reg,"\b"
  ret

' Code to Select Screen: Welcome
LBL Welcome
  PRINT2 "\e0G"
  ret

' Code to Select Screen: Length
LBL Length
  PRINT2 "\e91G"
  ret

' Code to Select Screen: Pieces
LBL Pieces
```

Figure 4.6: The LYNX Code Window

NOTE: The LYNX code window will be available after an update of the HMI or a separate compile. If a project is changed the code window must be closed and re-opened to update it.

The Notes Window

The notes window is a convenient place to take notes concerning the project. The contents of the notes window is automatically saved as "<project filename> Notes.txt" in the project directory. It may be printed by selecting "File>Print>Note Window" on the main window menu bar. The notes window may be viewed by selecting "View>Note Window".



```
*****HMI REGISTER DEMO*****
The purpose of this demo is to illustrate
registers and function keys being used
to effect events in a LYNX program.
The program used is a feed-to-length
application.

****Screens Needed****
Welcome
Length
Pieces
```

Figure 4.7: The LYNX Notes Window

Putting It All Together - Building Projects

Components of an HMI Project

The HMI ScreenBuilder is a project/screen based configuration utility for the HMI. What this means is that the user will begin by creating a project. Within this project the user will create as many HMI screens as are needed for the project. Registers and function key settings will be tied to the screens to which they are applied. Note that the screens, while numbered 1 . . .2. . .X, they are not necessarily sequential. They may be named and ordered in any fashion.

The project hierarchy is shown in figure 4.8.

Planning the HMI ScreenBuilder Project

In any project, planning is an important primary step. In most situations the HMI will be used in tandem with a LYNX controller product. Thus it is important to know exactly what needs to be accomplished by both parts of the system: the HMI and the LYNX. There are three basic ways the HMI may be used in a system to interact with the LYNX:

- To set variables, flag states, issue immediate mode instructions or display data outside a LYNX program (HMI Controls LYNX).
- To affect events inside a LYNX program (LYNX Controls HMI).
- A combination of the above two.

In each instance the HMI ScreenBuilder can be used to configure the system. However, there are several things that will need to be noted as certain HMI ScreenBuilder commands and features will operate differently between LYNX immediate mode and program mode. These differences will be noted in the subsections pertaining to a particular feature or command.

The flowchart in figure 4.9 illustrates the step-by-step flow of a project created using the HMI ScreenBuilder. This can be used to pre-plan the flow of the screens in your project. This flowchart can be a valuable resource in planning your HMI ScreenBuilder project.

Be aware that the HMI ScreenBuilder represents only half of the total programming that will need to be done. It will not create a completed LYNX program for you. The only LYNX code generated by the HMI will be a series of subroutines that access areas of HMI memory where screen and register data is stored. If the HMI is interacting with the LYNX controller product to control events within a program, these subroutines will have to be called using the "CALL" instruction to display screens or enter data into a register.

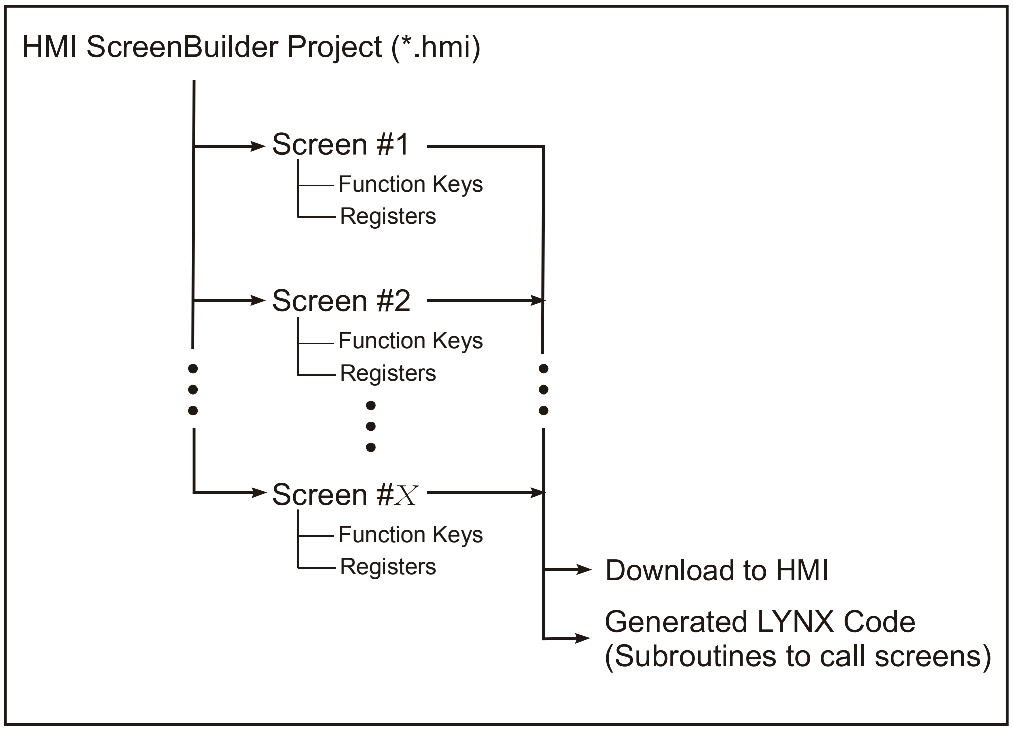


Figure 4.8: HMI Project Hierarchy

While LYNX programming is beyond the scope of this document, there are three sample projecst and associated LYNX programs to illustrate the interaction between the HMI and the LYNX controller product. The additional projects may also be loaded from the "Samples" folder of the HMI ScreenBuilder installation directory.

Prior to creating an HMI ScreenBuilder project you should take a few moments to plan your project. You can do this in the form of a flowchart, or by simply answering the following questions on a piece of paper:

- What screens will be required in the project?
- Will function keys be used to affect my final LYNX program?
- Will registers be used to transmit and receive data?
- Which screens will require the use of function keys and/or registers?

Creating a New Project

Upon start-up, the HMI ScreenBuilder will offer the option of opening an existing project, or creating a new project. To create a new HMI Project with the HMI ScreenBuilder open, click "File>New Project" on the menu bar.

Please note that this plan only needs to be a rough outline. The HMI ScreenBuilder allows for the deletion of uneeded functions, registers or screens. Also remember that the screens do not have to be created sequentially, they may be called as needed.

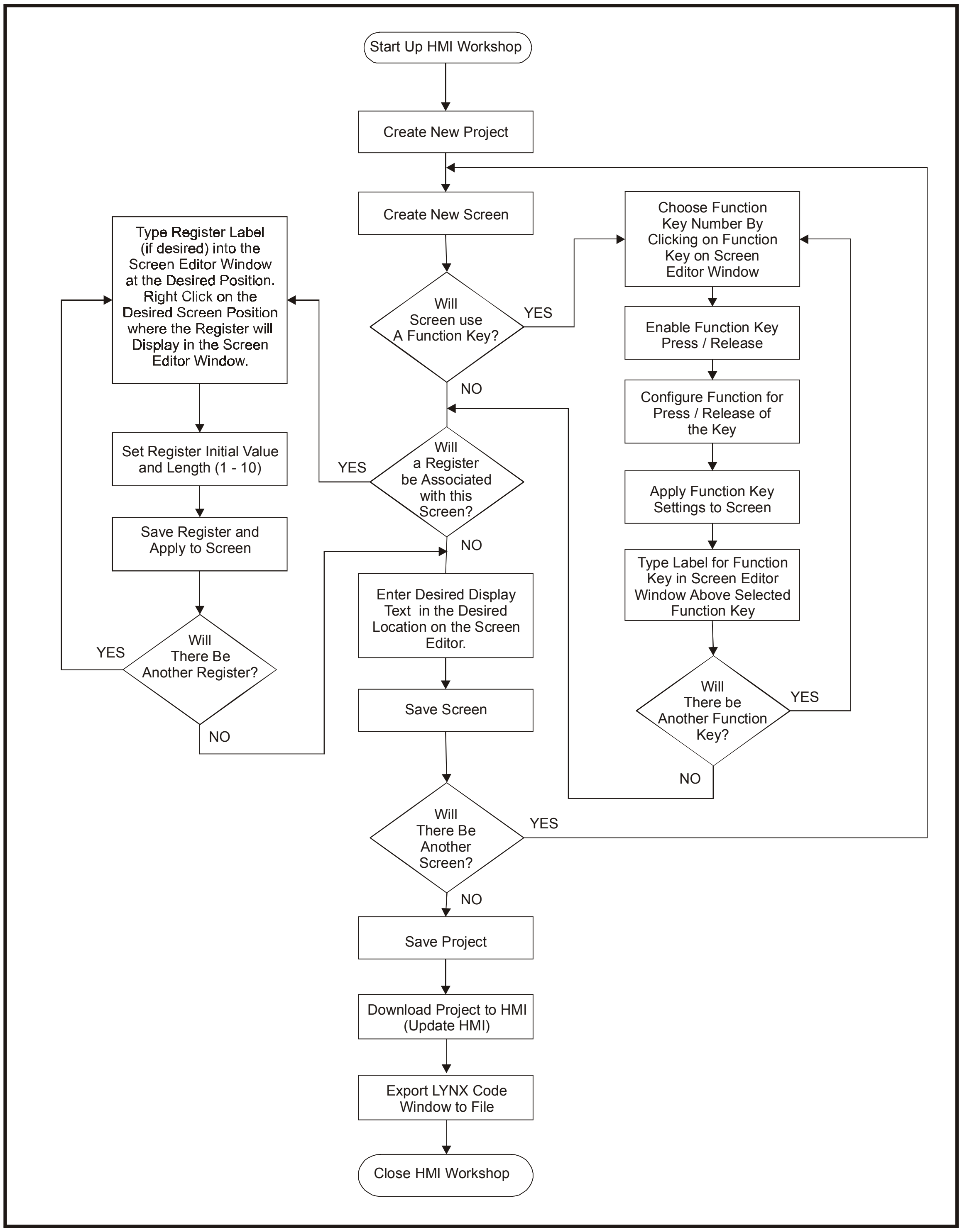

Start Up HMI Workshop
Create New Project
Create New Screen
Will Screen use A Function Key?
YES
NO
Choose Function Key Number By Clicking on Function Key on Screen Editor Window
Enable Function Key Press / Release
Configure Function for Press / Release of the Key
Apply Function Key Settings to Screen
Type Label for Function Key in Screen Editor Window Above Selected Function Key
Will There be Another Function Key?
YES
NO
Will a Register be Associated with this Screen?
YES
NO
Type Register Label (if desired) into the Screen Editor Window at the Desired Position. Right Click on the Desired Screen Position where the Register will Display in the Screen Editor Window.
Set Register Initial Value and Length (1 - 10)
Save Register and Apply to Screen
Will There Be Another Register?
YES
NO
Enter Desired Display Text in the Desired Location on the Screen Editor.
Save Screen
Will There Be Another Screen?
YES
NO
Save Project
Download Project to HMI (Update HMI)
Export LYNX Code Window to File
Close HMI Workshop

Figure 4.9: HMI Project Flow

NOTE: This manual only steps you through one complete project, additional sample projects are available in the "Samples" folder of the HMI ScreenBuilder installation directory!

What Goes Where?

Your project will likely have two distinct parts: The HMI ScreenBuilder project and the LYNX program code.

The ScreenBuilder portion of the project will be set up first, and be completed prior to any LYNX programming being done. While a precise understanding of what is going on in the background of the ScreenBuilder GUI is not necessary to use the HMI, it is good to understand the concepts of what is happening behind the scenes.

Escape What?

The HMI understands a collection of escape codes (Appendix A of this document covers these in detail). These escape codes control all of the HMI's functions such as storing screen text, register data, function key settings and setup parameters. Writing a program using purely escape codes is possible and has been done by IMS engineering, but it is time consuming and requires extensive programming skills. Thus the ScreenBuilder utility, which gives you a simple, intuitive interface which translates your intentions into these escape codes.

The ScreenBuilder utility is actually doing two things for you. First: it generates those escape codes which represent your screen content, registers, function key settings and associated text strings, and stores them in HMI memory when you click "Project>Update HMI". Second: is that the screen builder generates LYNX program code which forms the skeleton of your final LYNX program.

Each screen and register is treated as a LYNX program subroutine. These subroutines contain escape codes which are pointers to the locations in HMI memory where screen text and register data is stored. When CALL'ed, these subroutines will access those HMI memory locations and perform the actions dictated. Figure 4.10 below illustrates the system interaction between the HMI ScreenBuilder, LYNX Terminal, the HMI and the LYNX Controller product.

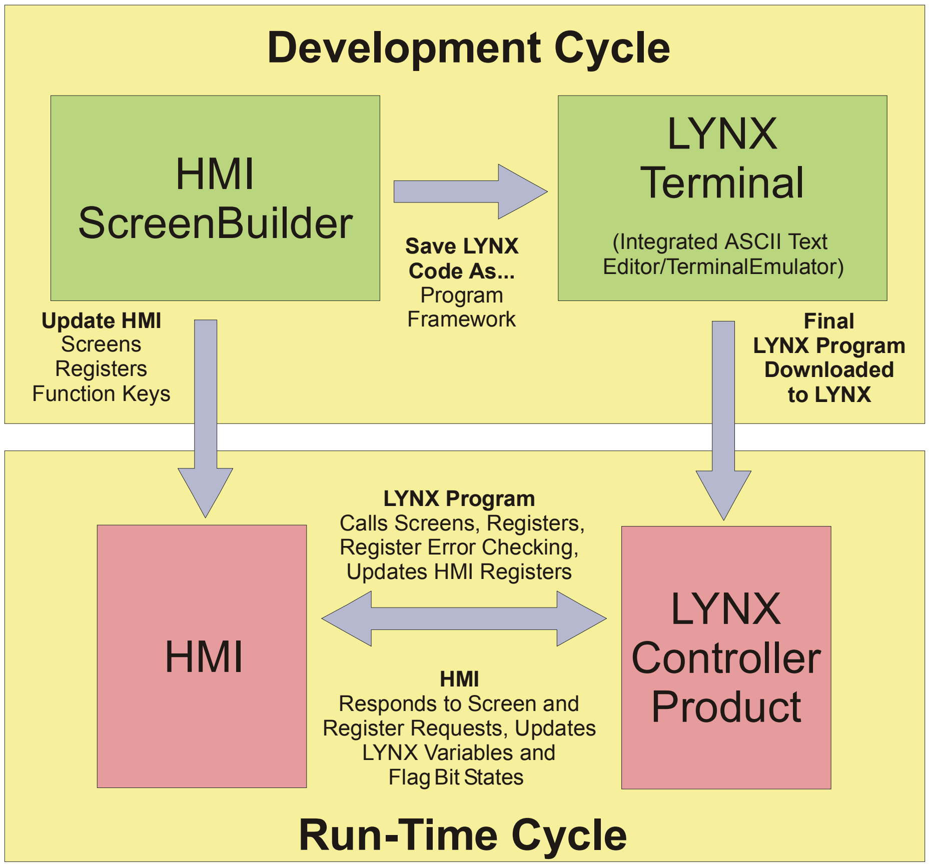

Development Cycle
HMI ScreenBuilder
LYNX Terminal
(Integrated ASCII Text Editor/TerminalEmulator)
Save LYNX Code As... Program Framework
Update HMI Screens Registers Function Keys
Final LYNX Program Downloaded to LYNX
HMI
LYNX Controller Product
LYNX Program Calls Screens, Registers, Register Error Checking, Updates HMI Registers
HMI Responds to Screen and Register Requests, Updates LYNX Variables and Flag Bit States
Run-Time Cycle

Figure 4.10: HMI - LYNX Interaction

Sample Projects

This subsection contains three short and simple projects designed to aquaint you with not only the HMI ScreenBuilder utility, but also developing your LYNX program that will interact with the HMI.

LYNX programming itself is outside the scope of this document. If you are unfamiliar with the LYNX programming language you may want to review the MicroLYNX QuickManual and/or the LYNX Family Product Manual Part 3: Software Reference. Both of these are available in Adobe Acrobat PDF format on the CD that was included with your HMI. You may desire to have either or both of these documents handy as you go through the sample projects shown.

The projects will cover two different areas:

- Project 1: HMI registers will be used to affect LYNX variables.
- Project 2: Similar to Project 1 in that HMI registers will affect LYNX variables, but the LYNX will update HMI register values as the program runs.
- Project 3: A party mode project where the HMI controls two axes of motion.

Tools and Equipment Required

Project 1

1] An HMI.
2] A MicroLYNX (or LYNX Control Module and Driver).
3] A motor properly sized for the MicroLYNX/Drive used.
4] A power supply sized for the system hardware being used.
5] HMI ScreenBuilder installed on a PC running Windows 98/NT4/2000/ME.
6] LYNX Terminal installed on a PC running Windows 98/NT4/2000/ME.
7] Free COM Port on PC.
8] Communications Cable MX-CC400-000 or Equivalent.

Project 2

1] An HMI.
2] 2 MicroLYNX Systems (or 2 LYNX Control Modules and Drivers).
3] A motor properly sized for the MicroLYNXes/Drives used.
4] A power supply sized for the system hardware being used.
5] HMI ScreenBuilder installed on a PC running Windows 98/NT4/2000/ME.
6] LYNX Terminal installed on a PC running Windows 98/NT4/2000/ME.
7] Free COM Port on PC.
8] Communications Cable MX-CC300-000 or Equivalent.

Sample Project #1:

This project will use data entered into HMI registers to update variables used in a LYNX program. There are three screens: a welcome screen, a setup screen and a run screen. This project will emulate an application where the end-user needs to specify a speed and distance for a move. Error checking prohibits the user from entering out-of-range values.

The project and associated LYNX program also reside in completed form in the “Samples Folder” of your HMI ScreenBuilder installation directory. The file name is sample_project_1.hmi, and the filename for the LYNX program is sample_project_1.lxt.

Starting the Project

Open the HMI ScreenBuilder by clicking "Start>Programs>HMI ScreenBuilder>HMI ScreenBuilder". In the project dialog, type the following: "_sample_project_1". The project filename will automatically set itself to the project name.

Create Screen 1

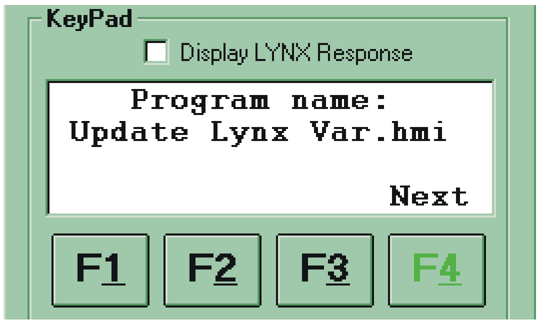
Screen #1 is the welcome screen.

Setup Screen

1] In the HMI setup screen enter "Screen1" in the "Save As..." text box.

2] Enter the text into the display window as shown in figure 4.11.

3] DO NOT check the "Start Up Screen" box. The welcome screen will be called up by the LYNX program.



KeyPad

☐ Display LYNX Response

```
  Program name:
Update Lynx Var.hmi

                Next
```

Figure 4.11: Screen 1 Setup

Setup Function Key F4

In this screen, the F4 key will advance the HMI display to the next screen.

1] Open the Function key dialog by clicking F4 on the HMI setup window.

2] Enter "Screen2" in the Press Text Box.

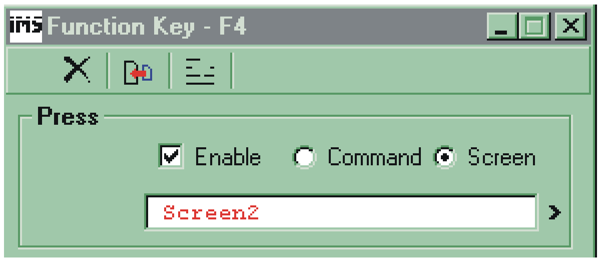
3] Click the "Screen" option.

4] Check the "Enable" box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



Function Key - F4

Press

☑ Enable ○ Command ◉ Screen

Screen2

Figure 4.12: Screen 1 Function Key F4 Setup

Usage Note! The names given to screens and registers are case sensitive. Each Screen name MUST be entered exactly as it is given.

Create Screen 2

Screen #2 will prompt the user to enter the piece length in inches. This value will be stored in a LYNX variable labeled “L_Reg”.

Setup Screen

1] In the HMI setup screen select <new> in the “Load Screen Box”, then enter “Screen2” in the “Save As...” text box.

2] Enter the text into the display window as shown in figure 4.13.

Setup the Distance Register

1] Position the mouse cursor in the screen row/column position directly to the right of the colon following “Ent Dist”, right click. Select “Insert Register”.

2] In the Configure Register dialog (figure 4.14), type “dist” in the “Name” box.

3] Set the length of the register to 2 digits.

4] Save register, apply to screen.

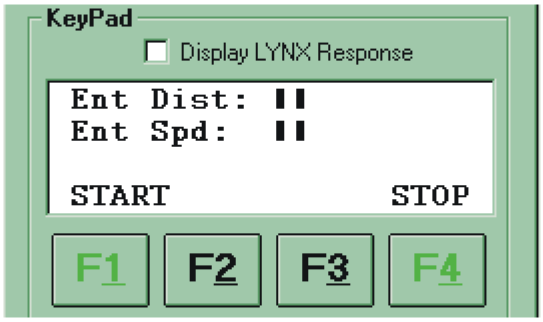
Setup the Speed Register

1] Position the mouse cursor in the screen row/column position directly to the right of the colon following “Ent Spd”, right click. Select “Insert Register”.

2] In the Configure Register dialog (figure 4.15), type “spd” in the “Name” box.

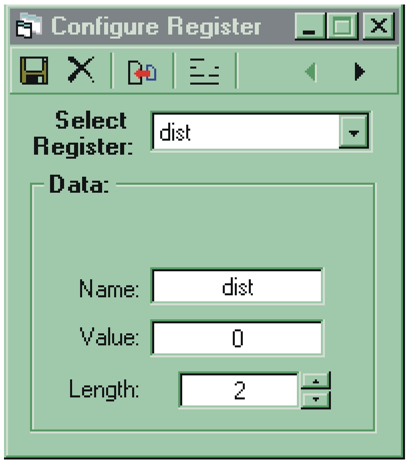
3] Set the length of the register to 2 digits.

4] Save register, apply to screen.



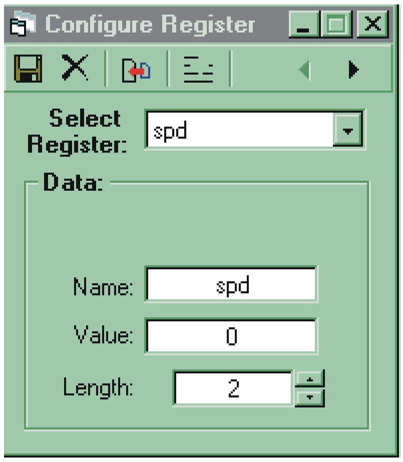
```
Ent Dist: ▮▮
Ent Spd:  ▮▮

START              STOP
```

Figure 4.13: Screen 2 Setup



Select Register: dist
Name: dist
Value: 0
Length: 2

Figure 4.14: Screen 2 Distance Register Setup



Select Register: spd
Name: spd
Value: 0
Length: 2

Figure 4.15: Screen 2 Speed Register Setup

Setup Function Key F1

In this screen, the F1 key will send a label down to the LYNX Controller product to start the program, which in turn will retreive register values from the HMI and process them.

1] Click the "F1" key on the HMI setup window. The Function Key-F1 setup dialog will open.

2] Type the word "Start" into the Press text box.

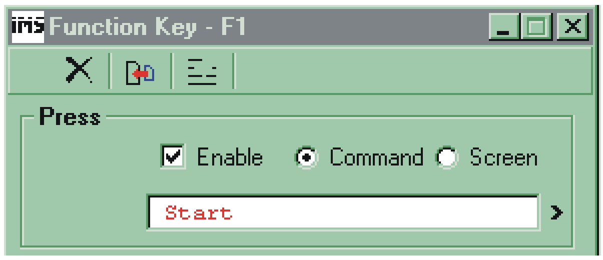
3] Enable the "Command" option.

4] Check the "Enable" box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



[✓] Enable [✓ Command] Screen

Start

Figure 4.16: Screen 2 Function Key F1 Setup

Setup Function Key F4

In this screen, the F1 key will send a label down to the LYNX Controller product to start the program, which in turn will retreive register values from the HMI and process them.

1] The setup dialog for F4 should already be open. Click the arrow to the right of the text box, this will open the command worksheet. Figure 4.18

2] On the "INST" pull-down select the LYNX soft-stop command (SSTP).

3] Type the number "0" following the SSTP entry.

4] Click the "Accept" button.

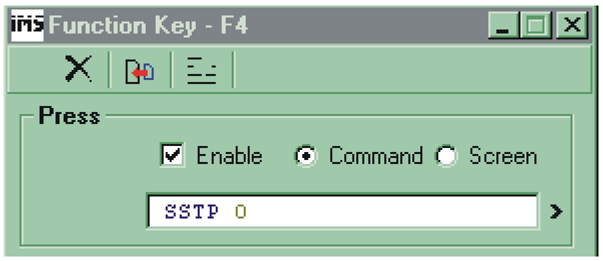
5] Enable the "Command" option.

6] Check the "Enable" box.

7] Apply the function key to the screen.

8] Save the screen.

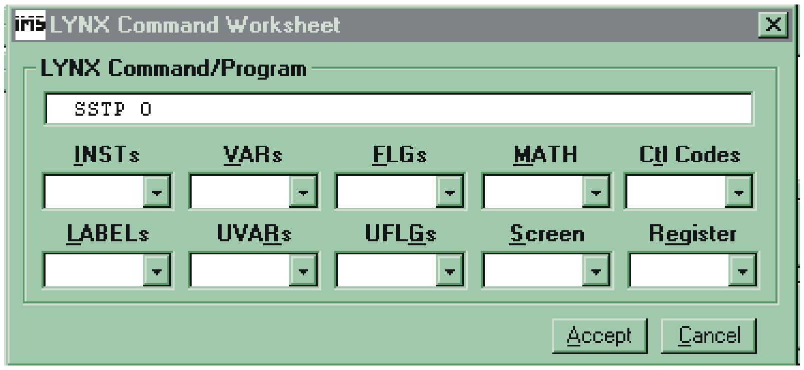
9] Save the project.



[✓] Enable [✓ Command] Screen

SSTP 0

Figure 4.17: Screen 2 Function Key F4 Setup



SSTP 0

Figure 4.18: LYNX Command Worksheet

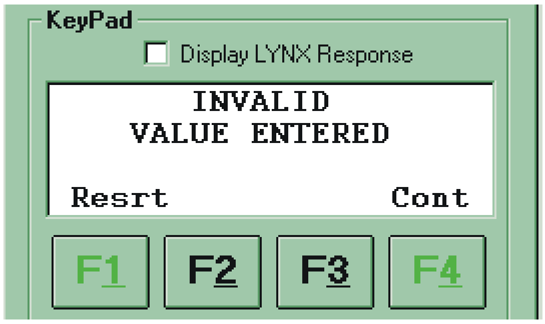
Create Screen

This screen will display if an invalid register value is entered. Screen 3 will be called if the value of either register is out of range.

Setup Screen

1] In the HMI setup screen enter "Screen3" in the "Save As..." text box.

2] Enter the text into the display window as shown in figure 4.19.



INVALID
VALUE ENTERED
Resrt Cont

Figure 4.19: Screen 3 Setup

Setup Function Key F1

In this screen, the F1 key will return the HMI to Screen 1.

1] The setup dialog for F1 should already be open.

2] Enter "Screen1" in the Press Text Box.

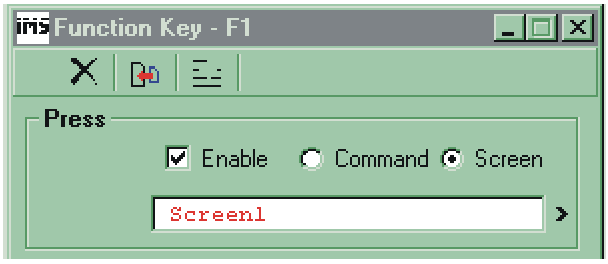
3] Click the "Screen" option.

4] Check the "Enable" box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



[✓] Enable ○ Command [● Screen]

Screen1

Figure 4.20: Screen 3 Function Key F1 Setup

Setup Function Key F4

In this screen, the F4 key will return the HMI to Screen 2.

1] The setup dialog for F4 should already be open.

2] Enter "Screen2" in the Press Text Box.

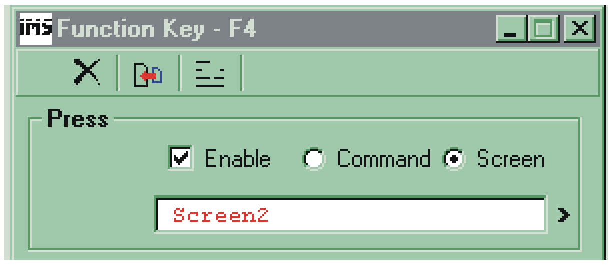
3] Click the "Screen" option.

4] Check the "Enable" box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



[✓] Enable ○ Command [● Screen]

Screen2

Figure 4.21: Screen 3 Function Key F4 Setup

Finishing the HMI ScreenBuilder Project

1] Click “Project>Update HMI>(method depends upon connection)”

2] Click File “Save LYNX Code As”

3] Save the Project

4] Exit ScreenBuilder

The LYNX Program

Open the LYNX code file in the LYNX Terminal Text Editor window.

The program is shown below. The highlighted areas represent the portion of the program that you will write. These blocks of code will enable the LYNX controller product to interact with the HMI.

The program actually consists of two programs labeled “startup” and “start”. “Startup” will run upon system power-on. This program will call screen1, the welcome screen, to the HMI display and then end. By pressing the F4 key on the HMI the display will advance to screen2, the entry screen.

The user will enter the desired distance and speed into the HMI registers and press the F1 key, which has been programmed to send the “Start” label down to the LYNX product. This will execute the second program. This program will first verify that the two registers used for distance and speed are within the specified range. If the registers are in the specified range, the program will execute the move the distance and speed specified. Following the completion of the move, the program will call up screen 1 and end.

If either of the registers are out of the specified range, screen 3 will be called up and the program will end. At this point the user has the option of using the two programmed function keys to either return to screen 1 or screen 2.

```
' Start of HMI Register declarations
VAR dist = 0                      ' register length = 2
VAR spd = 0                       ' register length = 2

' Start of LYNX VAR/FLG declarations
Munit=51200                       'set motor units to 51,200 msteps/rev
Mac=100                           'set acceleration current to 100%
Mrc=75                            'set motor run current to 75%
Mhc=0                             'set motor holding current to 0%

' **** Start of LYNX Code ****
Pgm 10                            'enter program mode,start progam at lynx address 10
LBL startup                       'label program to initiate on power-up
     CALL screen1                 'call subroutine for screen1
END                               'end the program
Pgm                               'exit program mode

Pgm 100                           'enter program mode,start program at lynx address 100
LBL start                         'label the program start [runs when F4 is pressed]
     CALL mxmn_0                  'call error checking subroutine for dist register
     CALL mxmn_1                  'call error checking subroutine for spd register
     Vm=spd                       'set maximum velocity equal to spd register
     MOVR dist                    'relative index amount specified by dist register
     HOLD 2                       'suspend program execution until motion completes
     CALL screen1                 'call subroutine for screen1
END                               'end the program
```

```
'The following HMI ScreenBuilder generated subroutines have been modified for
'ease of viewing, the commented lines have been changed, the generated
'LYNX code has only been modified to allow for register error checking

' Code to Fill Registers
LBL REG_0
     Pfmt = 2,0,2
     PRINT2 "\e0v",dist,"\b"
  RET

LBL REG_1
     Pfmt = 2,0,2
     PRINT2 "\e1v",spd,"\b"
  RET

' Code to Test Registers
LBL MxMn_0
     BR FAIL_0, dist >= 100
     BR FAIL_0, dist <= -10
  RET
LBL FAIL_0
     dist = 0
     CALL REG_0
     CALL screen3                'call screen3 upon failure
LBL OK_0
  RET

LBL MxMn_1
     BR FAIL_1, spd >= 100
     BR FAIL_1, spd <= -10
  RET
LBL FAIL_1
     spd = 0
     CALL REG_1
     CALL screen3                'call screen3 upon failure
LBL OK_1
  RET

' Code to Select Screens

LBL Screen1
     PRINT2 "\e0G"
  RET

LBL Screen2
     PRINT2 "\e85G"
  RET

LBL Screen3
     PRINT2 "\e228G"
  END

END
Pgm
SAVE
```

Sample Project #2:

Where Project #1 used data entered into HMI registers to update variables used in a LYNX program, this project will use data processed within a LYNX program to update HMI register values. There are two screens: a setup screen and a run screen. This project will emulate a feed to length application where the end-user needs to specify a piece length and quantity. The run screen will show the quantity entered and the quantity completed.

This project and associated LYNX program also reside in completed form in the "Samples Folder" of your HMI ScreenBuilder installation directory. The file name is sample_project_2.hmi, and the filename for the LYNX program is sample_project_2.lxt.

Starting the Project

Open the HMI ScreenBuilder by clicking "Start>Programs>HMI ScreenBuilder>HMI ScreenBuilder". In the project dialog, type the following: "_sample_project_2". The project filename will automatically set itself to the project name.

Create Screen 1

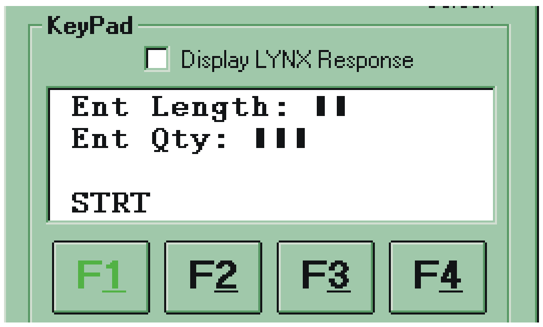
Screen #1 is the data entry screen.

Setup Screen

1] In the HMI setup screen enter "Screen1" in the "Save As..." text box.

2] Enter the text into the display window as shown in figure 4.11.

3] DO NOT check the "Start Up Screen" box. The welcome screen will be called up by the LYNX program.



```
Ent Length: ▮▮
Ent Qty: ▮▮▮

STRT
```

Figure 4.22: Screen 1 Setup

Setup Function Key F1

In this screen, the F1 key will send a label to the LYNX product to start the program.

1] Open the Function key dialog by clicking F1on the HMI setup window.

2] Enter "start" in the Press Text Box.

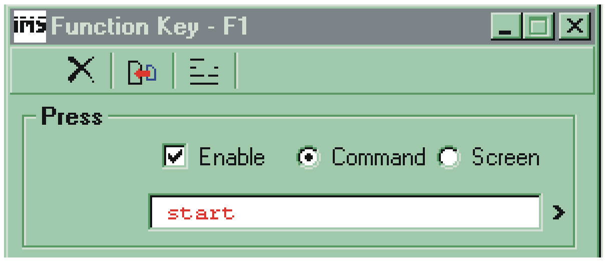
3] Click the "Command" option.

4] Check the "Enable" box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



Press

[✓] Enable [● Command] ○ Screen

start

Figure 4.23: Screen 1 Function Key F1 Setup

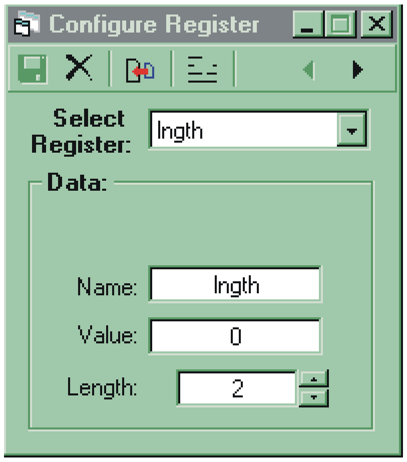
Setup the Length Register

1] Position the mouse cursor in the screen row/column position directly to the right of the colon following “Ent Length”, right click. Select “Insert Register”.

2] In the Configure Register dialog (figure 4.24), type “lngth” in the “Name” box.

3] Set the length of the register to 2 digits.

4] Save register, apply to screen.



Select Register: lngth
Name: lngth
Value: 0
Length: 2

Figure 4.24: Screen 1 Length Register Setup

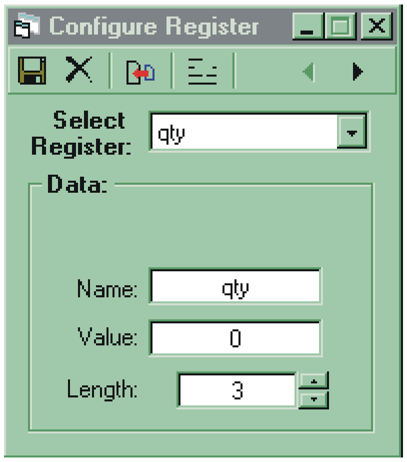
Setup the Quantity Register

1] Position the mouse cursor in the screen row/column position directly to the right of the colon following “Ent Qty”, right click. Select “Insert Register”.

2] In the Configure Register dialog (figure 4.25), type “qty” in the “Name” box.

3] Set the length of the register to 3 digits.

4] Save register, apply to screen.



Select Register: qty
Name: qty
Value: 0
Length: 3

Figure 4.25: Screen 1 Quantity Register Setup

Create Screen 2

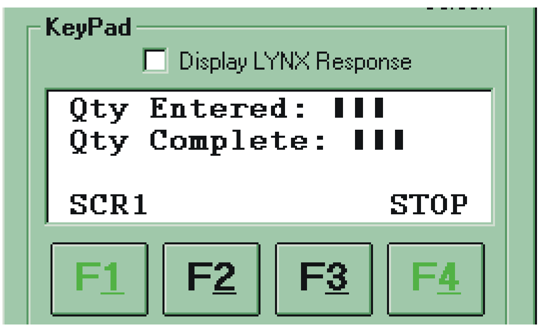
Screen #2 is the run screen.

Setup Screen

1] In the HMI setup screen enter “Screen2” in the “Save As...” text box.

2] Enter the text into the display window as shown in figure 4.26.

3] Insert the Quantity register as shown in figure 4.26. The Qty Complete register has not been created yet.



```
Qty Entered: ▮▮▮
Qty Complete: ▮▮▮

SCR1                STOP
```

Figure 4.26: Screen 2 Setup

Setup Function Key F1

In this screen, the F1 key will bring screen 1 to the display.

1] The Function key dialog for F1 should already be open.

2] Enter “Screen1” in the Press Text Box.

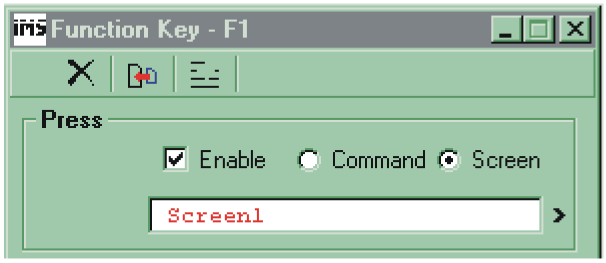
3] Click the “Screen” option.

4] Check the “Enable” box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



☑ Enable ○ Command [✓ Screen]

Screen1

Figure 4.27: Screen 2 Function Key F1 Setup

Setup Function Key F4

In this screen, the F4 key will issue the soft stop command to the LYNX product.

1] Open the Function key dialog by clicking F4 on the HMI setup window.

2] Enter “SSTP 1” in the Press Text Box.

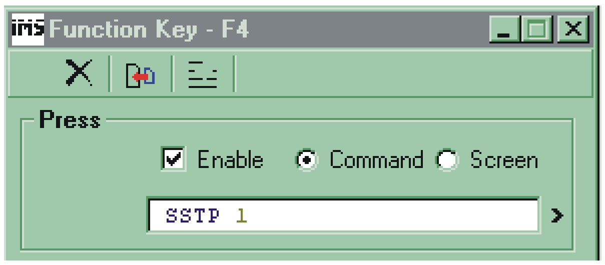
3] Click the “Command” option.

4] Check the “Enable” box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



☑ Enable [✓ Command] ○ Screen

SSTP 1

Figure 4.28: Screen 2 Function Key F4 Setup

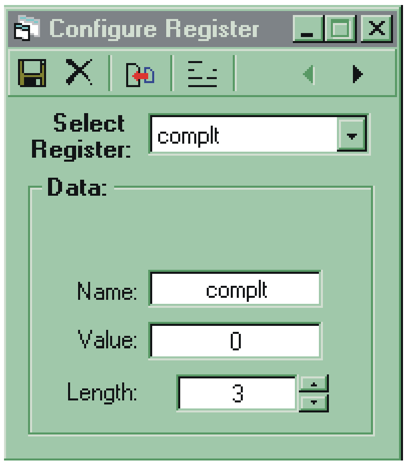
Setup the Quantity Complete Register

1] Position the mouse cursor in the screen row/column position directly to the right of the colon following “Qty Complete”, right click. Select “Insert Register”.

2] In the Configure Register dialog (figure 4.29), type “complt” in the “Name” box.

3] Set the length of the register to 3 digits.

4] Save register, apply to screen.



Select Register: complt

Name: complt

Value: 0

Length: 3

Figure 4.29: Screen 2 Quantity Complete Register Setup

Finishing the HMI ScreenBuilder Project

1] Click “Project>Update HMI>(method depends upon connection)”

2] Click File “Save LYNX Code As”

3] Save the Project

4] Exit ScreenBuilder

The LYNX Program

Open the LYNX code file in the LYNX Terminal Text Editor window.

The program is shown below, the highlighted areas represent the portion of the program that you will write. These blocks of code will cause the LYNX controller product to be able to interact with the HMI.

The user will enter the desired length and quantity into the HMI registers and press the F1 key, which has been programmed to send the “Start” label down to the LYNX product. This will execute the process which will index the length and number of times specified by the register entries. The LYNX program will increment the Quantity completed register and display on screen until the specified number of moves complete.

The user may then press F1 to return to screen 1 and restart the process. At any time during the moves the user may stop the program by pressing the F4 key.

```
' Start of HMI Register declarations
VAR lngth = 0              ' register length = 2
VAR qty = 0                ' register length = 3
VAR complt = 0             ' register length = 3
```

```
' Start of LYNX VAR/FLG declarations
Munit=51200                'set the motor units variable to 51200
Mac=100                    'set the acceleration current to 100%
Mrc=75                     'set the run current to 75%

' **** Start of LYNX Code ****
Pgm 10                     'enter program mode at address 10
LBL startup                'label the proram to execute upon power-up
   CALL screen1            'call screen 1
   END                     'end the program

LBL start                  'label process start
   CALL screen2            'call the run screen
   complt=0                'initialize global variable complt
   CALL reg_1              'call quantity register
   CALL reg_2              'call completed register
   LBL loop                'label sub-process loop
      MOVR lngth           'relative index amount spcfd by length register
      HOLD 2               'suspend program execution until motion completes
      INC complt           'increment the global variable complt
      CALL reg_2           'update completed register
      DELAY 500            'delay program execution for .5 seconds
      BR loop,complt < qty 'cond. branch to loop while complt is less than qty
    END                    'end program
```

```
' Code to Fill Registers
LBL REG_0
      Pfmt = 2,0,2
      PRINT2 "\e0v",lngth,"\b"
  RET

LBL REG_1
      Pfmt = 3,0,2
      PRINT2 "\e1v",qty,"\b"
  RET

LBL REG_2
      Pfmt = 3,0,2
      PRINT2 "\e2v",complt,"\b"
  RET


' Code to Test Registers
LBL MxMn_0
      BR FAIL_0, lngth >= 100
      BR FAIL_0, lngth <= -10
  RET
LBL FAIL_0
      lngth = 0
      CALL REG_0
LBL OK_0
  RET

LBL MxMn_1
      BR FAIL_1, qty >= 1000
      BR FAIL_1, qty <= -100
  RET
LBL FAIL_1
      qty = 0
      CALL REG_1
LBL OK_1
  RET

LBL MxMn_2
      BR FAIL_2, complt >= 1000
      BR FAIL_2, complt <= -100
  RET
LBL FAIL_2
      complt = 0
      CALL REG_2
LBL OK_2
  RET

' Code to Select Screens
LBL Screen1
      PRINT2 "\e0G"
  RET

LBL Screen2
```

```
          PRINT2 "\e123G"
     RET

   END
   Pgm
   SAVE
```

Sample Project #3:

Sample Project #3 uses two HMI registers to store and send the X and Y distance values to a dual-axis system using two MicroLYNXes. This project consists of two screens.

This project and associated LYNX programs also reside in completed form in the "Samples Folder" of your HMI ScreenBuilder installation directory. The file name is sample_project_3.hmi, and the filename for the LYNX programs are sample_project_3_1.lxt and sample_project_3_2.lxt.

Starting the Project

Open the HMI ScreenBuilder by clicking "Start>Programs>HMI ScreenBuilder>HMI ScreenBuilder". In the project dialog, type the following: "_sample_project_3". The project filename will automatically set itself to the project name.

Create Screen 1

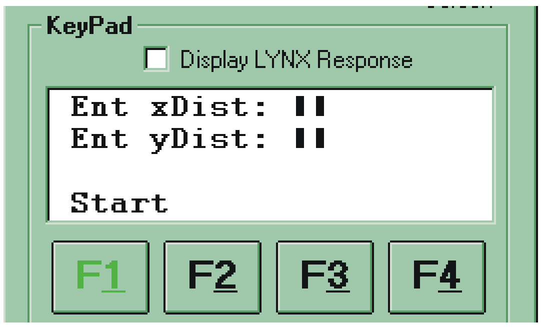
Screen #1 is the data entry screen. On this screen the two registers will be entered which will control the X and Y distance moved by each axis.

Setup Screen

1] In the HMI setup screen enter "Screen1" in the "Save As..." text box.

2] Enter the text into the display window as shown in Figure 4.30.

3] DO NOT check the "Start Up Screen" box. The welcome screen will be called up by the LYNX program.



```
Ent xDist: ▌▌
Ent yDist: ▌▌

Start
```

Figure 4.30: Screen 1 Setup

Setup Function Key F1

In this screen, the F1 key will send a label to MicroLYNX 1 to start the program.

1] Open the Function key dialog by clicking F1 on the HMI setup window.

2] Enter "Start" in the Press Text Box.

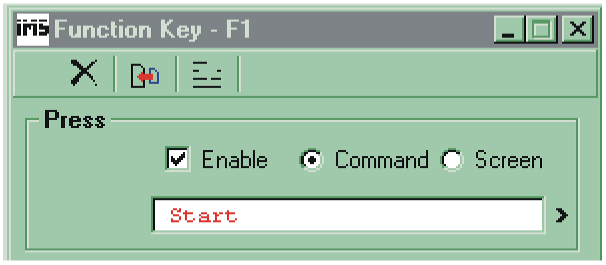
3] Click the "Command" option.

4] Check the "Enable" box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



☑ Enable (● Command) ○ Screen

Start

Figure 4.31: Screen 1 Function Key F1 Setup

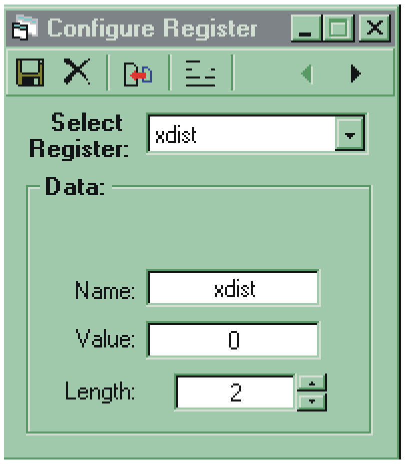
Setup the X Distance Register

1] Position the mouse cursor in the screen row/column position directly to the right of the colon following "xDist", right click. Select "Insert Register".

2] In the Configure Register dialog (Figure 4.32), type "xdist" in the "Name" box.

3] Set the length of the register to 2 digits.

4] Save register, apply to screen.



Select Register: xdist

Name: xdist

Value: 0

Length: 2

Figure 4.32: Screen 1 X Distance Register Setup

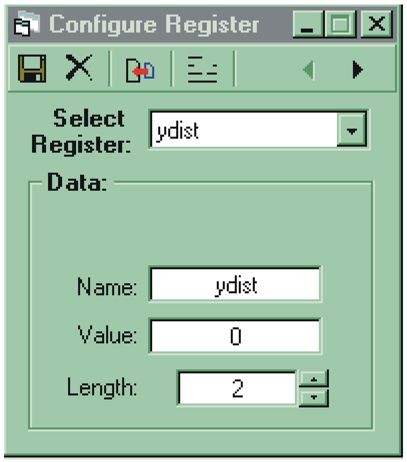
Setup the Y Distance Register

1] Position the mouse cursor in the screen row/column position directly to the right of the colon following "yDist", right click. Select "Insert Register".

2] In the Configure Register dialog (figure 4.33), type "ydist" in the "Name" box.

3] Set the length of the register to 3 digits.

4] Save register, apply to screen.



Select Register: ydist

Name: ydist

Value: 0

Length: 2

Figure 4.33: Screen 1 Y Distance Register Setup

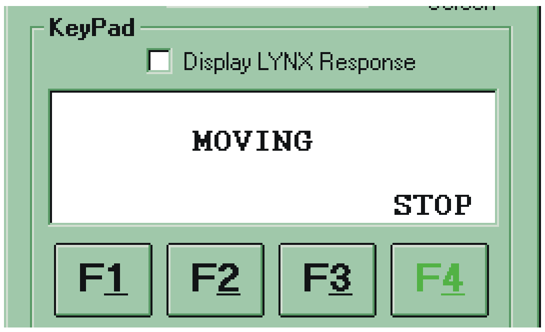
Create Screen 2

Screen #2 will be visible when the axes are moving, the F4 key will give the user the option of stopping the motion.

Setup Screen

1] In the HMI setup screen enter "Screen2" in the "Save As..." text box.

2] Enter the text into the display window as shown in Figure 4.34.



```
MOVING

                STOP
```

Figure 4.34: Screen 2 Setup

Setup Function Key F4

In this screen, the F4 key will issue a Soft Stop command to the system when pressed.

1] Open the Function key dialog by clicking F4 on the HMI setup window.

2] Enter "SSTP 0" in the Press Text Box.

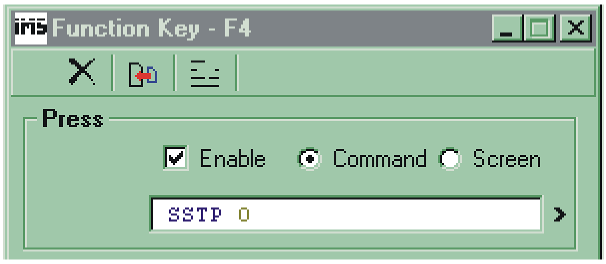
3] Click the "Command" option.

4] Check the "Enable" box.

5] Apply the function key to the screen.

6] Save the screen.

7] Save the project.



[✓ Enable] [✓ Command] Screen

SSTP 0

Figure 4.35: Screen 2 Function Key F4 Setup

Finish the Project

1] Update the HMI

2] Save LYNX Code As. . .[project filename]

The LYNX Programs

Open the LYNX code files in the LYNX Terminal Text Editor window.

There are two separate programs for each of the MicroLYNXes in the system. The HOST MicroLYNX, MicroLYNX 1, needs to be given the node address "A" using the command DN=A in the LYNX Terminal. This is the MicroLYNX which will have subroutines to call screens and registers.

Each program will have a process with the label "start". The press of function key F1 on screen 1 will execute the process start in the MicroLYNX #1 program. MicroLYNX #1 will retrieve the register values then forward the Y distance value to MicroLYNX #2, as well as the "start" label, thus executing the program in MicroLYNX #2.

Screen 2 will then be called to the display, MicroLYNX #1 will move X distance, and wait until MicroLYNX #2 completes the Y distance move. When MicroLYNX #2 completes the Y distance move it will set a flag bit and print that state to the RS-485 buss. When MicroLYNX #1 sees that bit set it will call up Screen 1 to the HMI display.

As with the previous two projects the highlighted code area is entered by the user, un-highlighted code is generated by the HMI ScreenBuilder. ScreenBuilder generated code has been edited for simplification.

Program for MicroLYNX #1

```
' Start of HMI Register declarations
VAR xdist = 0                            ' register length = 2
VAR ydist = 0                            ' register length = 2

' Start of LYNX VAR/FLG declarations
Mac=100                                  'motor acceleration current =100%
Mrc=75                                   'motor run current =75%
Munit=51200                              'motor unit variable = 51200
FLG done                                 'declare global user flag done

' **** Start of LYNX Code ****
Pgm 10                                   'Enter program mode at address 10
LBL STARTUP                              'label program to commence on power-up
  CALL Screen1                           'call HMI screen 1 to the display
  END                                    'End program

LBL start                                'label process start
  done=0                                 'set flag done to 0
  PRINT2 "\nBydist=",ydist               'print var ydist to lx comm2
  CALL screen2`                          'call screen 2 to hmi display
  MOVR xdist                             'index relative to current pos xdist
  PRINT2 "\nBstart"                      'post label “start” to lx comm2
  LBL wait                               'declare sub-process wait
    BR wait,done=0                       'loop to wait while done=0
    CALL screen1                         'call screen 1 when done=1
    END                                  'End program

' Code to Fill Registers
LBL REG_0
    Pfmt = 2,0,2
    PRINT2 "H\e0v",xdist,"\b"
  RET

LBL REG_1
    Pfmt = 2,0,2
    PRINT2 "H\e1v",ydist,"\b"
  RET

' Code to Test Registers
LBL MxMn_0
    BR FAIL_0, xdist >= 100
    BR FAIL_0, xdist <= -10
  RET
LBL FAIL_0
    xdist = 0
    CALL REG_0
```

```
LBL OK_0
  RET

LBL MxMn_1
    BR FAIL_1, ydist >= 100
    BR FAIL_1, ydist <= -10
  RET
LBL FAIL_1
    ydist = 0
    CALL REG_1
LBL OK_1
  RET

' Code to Select Screens
LBL Screen1
    PRINT2 "\nH\e0G"
  RET

LBL Screen2
    PRINT2 "\nH\e135G"
  RET
```

```
END
Pgm
SAVE
```

Program for MicroLYNX #2

```
'Program for Drive two of Party mode application.

' Start of LYNX VAR/FLG declarations
Mac=100                                  'motor acceleration current =100%
Mrc=75                                   'motor run current =75%
Munit=51200                              'motor unit variable = 51200
VAR ydist                        'declare user variable ydist

' **** Start of LYNX Code ****
Pgm 10                           'enter program mode at address 10
LBL start                        'label program start
  MOVR ydist                     'index relative to current pos ydist
  HOLD 2                         'suspend program execution until motion completes
  PRINT "\nAdone=1"              'set done flag bit state on drive A to 1
END                              'end program
Pgm                              'exit program mode
```

APPENDIX A

Escape Codes

Section Overview

This section covers the escape codes used in programming the HMI. Once again we stress that the easiest, most effective method of programming the HMI is to use the provided HMI ScreenBuilder. A thorough understanding of escape codes is recommended if the user intends to use them to program the HMI.

- Command Codes
- Escape Codes

ANSI Command Codes

The command codes are used by the HMI to assign a function or set a parameter. Three command codes are used. The command code most used will be the escape code <ESC>, or "\e". This code will always precede the code which will instruct the HMI to set a condition.

HMI Command Codes		
Code	**Character**	**Hex Value**
\e	Escape	1B
\t	Tab	09
\b	Backspace	08

Table A.1: HMI ANSI Codes

Escape Codes

All of these commands are prefixed by the ESC character. The LYNX will output the ESC when used with the PRINT instructions thus:

```
PRINT "\e<parameter><code>"
```

MOVE CURSOR

Function	Move Cursor
Code	<row>,<col>C
Description	
Move cursor to row <row>, column <col> <row>= 1 - 4 <col>= 1 - 20	
Example	
Move cursor to row 2 column 4 PRINT "\e2,4C"	

The “Move Cursor” command is used to position the cursor on the HMI display. This may be used to place the cursor to a position prior to printing a text string to the display as in the example below:

PRINT “\e2,7CWELCOME”

This would display the word “WELCOME” beginning at row 2, column 7 of the display.

See the figure below for the row/column map of the HMI display.

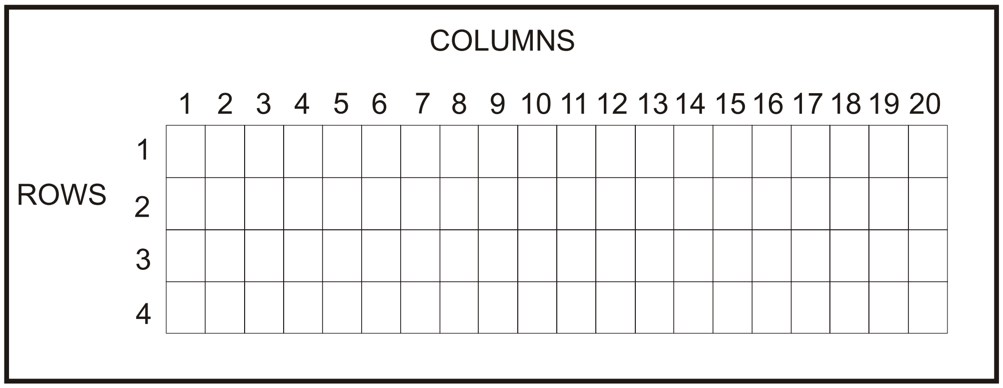


Figure A.1: Row/Column Map of the HMI Display

ERASE DISPLAY AND CURSOR HOME

Function	Erase Display and Cursor Home
Code	0E
Description	
Erase the entire display and return the cursor to the home position.	
Example	
print "\e0E"	

The “Erase Display and Cursor Home” command will erase any text displayed on the HMI display and return the cursor to the default, row 1/column 1 position.

This is especially useful in clearing a screen prior to moving to the next screen. Please note that any function key or register values are not effected by this command. However, if you have placed a text label above a function key the label will have to be reprinted on the next screen if the function of the key will be repeated in the LYNX program.

ERASE TO END OF LINE

The “Erase to End of Line” command will erase any text displayed on the HMI display from the current cursor position to the end of that row. The cursor position will not change.

This can be used to clear a screen prior to moving to a new screen where one or more lines of text on the display are to be retained, as in the case of function key labels on the 4th row.

In this case the command would be used in conjunction with the “Move Cursor” command.

The example code shown below would erase rows 2 and 3, leaving row 4 intact and return the cursor to its home position.

PRINT “\e2,1C\e0L\e3,1C\e0L\e1,1C”

Function	Erase to End of Line
Code	0L
Description	
Erase the display from cursor position to the end of the line.	
Example	
PRINT "\e0L"	

Usage Note! Multiple escape codes may be used on a single line of LYNX code. This allows the programmer to group operations in the HMI without having to enter a “PRINT” statement for each operation. See example for **ERASE TO END OF LINE.**

ERASE <X > NUMBER OF CHARACTERS

The “Erase <x> Number of Characters” command will erase a specified number of characters from the current position of the cursor. The main difference between this command and the other cursor control commands is that the cursor position will not change.

This is useful for erasing a string of text and replacing it with another string of text in the same display space. In the example we will first print the word “ERASE” at display row 1, column 1, then we will replace it with the word REPLACE.

PRINT “\e1,1CERASE”
PRINT “\e1,1C\e5BREPLACE”

As can be seen from the example it is not necessary to return the cursor to row 1 column 1 prior to printing the new word.

Function	Erase <num> Characters
Code	<num>B
Description	
Erase <num>(number of characters) from cursor position; cursor position does not change. <num>= 1 - 20	
Example	
Erase 15 characters from the current cursor position. PRINT "\e15B"	

Usage Note! Careful commenting of the escape codes entered will assist in speedy debugging of any typographic or code entry errors in your LYNX program or text file used to enter commands in immediate mode. Comment lines are always preceeded by the apostrophe (‘) character.

DISPLAY CONTROL

Function	Display Control
Code	<flag[2..0]>q
Description	
The flag # used will be the decimal equivalent of the binary number resultant from the desired setting. <flag2> is the most significant bit. <flag0>=1: Enable blinking cursor (default) <flag0>=0: Disable blinking cursor <flag1>=1: Cursor on (default) <flag1>=0: Cursor off <flag2>=1: Display on (default) <flag2>=0: Display off, LCD RAM is not cleared	
Example	
Set display on, cursor on, disable blinking cursor. PRINT "\e6q"	

The "Display Control" command will be the decimal equivalent of the three bit binary number resultant from the desired setting. The least significant bit is <flag 0>.

In the example we will change the display settings to turn off the blinking cursor. We will leave the other settings in the factory default state, thus:

<flag0> = 0	'Disable blinking cursor.
<flag1> = 1	'Cursor on.
<flag2> = 1	'Display on.

Binary 110 converted to decimal = 6, thus:

PRINT "\e6q"

disables the blinking cursor.

PRINT "\e7q" restores it.

ENABLE/DISABLE DISPLAY OF TEXT

Function	Enable/Disable Display of Text on the LCD
Code	<flag>d
Description	
The HMI will always respond to commands, this setting only disables the display of text. <flag>=1: Do not display text <flag>=0: Display text (default)	
Example	
Disable the display of text on the LCD. PRINT "\e1d"	

This command will enable or disable the display of text on the HMI LCD. This will not, however, affect the display of numeric register values.

This command will not defeat the HMI's ability to respond to commands, it will only affect the display of text.

This command is especially useful in the following type of scenario: You are using a function key to slew the motor at some velocity, but do not desire the LYNX command and the velocity to appear on the LCD because of LYNX echo.

User Output Escape Codes

USER OUTPUT ON/OFF

This “User Output ON/OFF” command is used to turn the user output (*See Section 2:Connecting the User Output*) on or off.

This command could be used in conjunction with a LYNX subroutine to turn on a siren or light when an error occurs, or a warning light when the machine is running.

Function	User Output ON/OFF
Code	<flag>O
Description	
<flag>=1: User output ON <flag>=0: User output OFF (default)	
Example	
Turn the user output ON. PRINT "\e1O"	

Data Entry/Response Escape Codes

WAIT FOR “NEXT” KEY PRESS

This command will halt the HMI program until the “NEXT” key is pressed.

An example use for this command is where a LYNX program completes an operation, then requests user acknowledgement before proceeding to the next step of the program.

See the sample program HOW_FAR, in Appendix B: Sample Programs for an example of how this command is used in this context.

Function	Wait for the Press of the NEXT Key
Code	0N
Example	
PRINT "\e0N"	

DELAY FOR 100 MILLISECONDS

This command will insert a 100ms delay between HMI operations. It can be used multiple times if a longer delay is needed. In the example code below we will clear the display, print some text to the screen, delay for 1 second, and print another line of text.

Function	Delay for 100 Milliseconds
Code	0D
Example	
PRINT "\eOD"	

```
PRINT "\e0EWait 1 Second\e0D\e0D\e0D\e0D\e0D\e0D\e0D\e0D\e0D\e0D\e2,1CWaited 1 Second"
```

HOLD DATA ENABLE/DISABLE

Function	Hold Data Enable/Disable
Code	<flag>H
Description	
<flag>=1: Data not transmitted after the ENTER key is pressed <flag>=0: Data transmitted after the ENTER key (default)	
Example	
Place HMI in hold data mode. PRINT "\e1H"	

This command will toggle the HMI in and out of the HMI in "Hold Data Mode" and would be used in conjunction with the "Release" command in a case where the data is being held.

This command is used in party mode to suspend the transmission of pressed keys.

By default, data will be transmitted by the HMI upon ENTER key press.

RELEASE HELD DATA

Function	Release
Code	0r
Description	
Transmit entered data that was held.	
Example	
Transmit held data. PRINT "\e0r"	

This command will transmit held data if the hold data mode remains the same.

CLEAR KEYPAD BUFFER

Function	Clear Keypad Buffer
Code	0K
Description	
Clears buffer holding keys pressed but not sent from the keypad.	
Example	
Clear keypad buffer. print "\e0K"	

This command will remove ALL held data from the keypad buffer.

General HMI Setup Escape Codes

These settings are stored in the non-volatile memory.

BAUD RATE

This command is used to set the communications BAUD rate of the HMI.

This setting MUST be consistent with the BAUD setting of the LYNX COMM port to which the HMI is connected.

The default setting for both the HMI and the LYNX controller products is 9600 bps.

Power must be cycled for a BAUD rate change to take affect.

Function	Set BAUD Rate
Code	<baud#>b
Description	
The power on the HMI must be cycled in order for change to take place. <baud#>= 4: 4800 <baud#>= 9: 9600 (default) <baud#>= 1: 19,200 <baud#>= 3: 38400	
Example	
Set the HMI communications BAUD rate to 19.2kbps. PRINT "\e1b"	

SET ECHO

LOCAL ECHO:
The HMI detects when the key is pressed and displays the character.

REMOTE ECHO:
HMI detects the key press and transmits the character. The character will be displayed when the remote device echoes the character.

Function	Set Echo
Code	<flag>e
Description	
<flag>= 0: Remote keypad echo <flag>= 1: Local keypad echo (default)	
Example	
Set remote echo. PRINT "\e0e"	

CARRIAGE CONTROL

Function	Carriage Control
Code	<flag>c
Description	
<flag>= 0: Do not respond to CR and LF (default) <flag>= 1: Respond to CR and LF	
Example	
Respond to CR and LF. PRINT "\e1c"	

This command will specify whether or not HMI will respond to a line feed or carriage return.

CONTROLLER PROMPT

Function	Controller Prompt
Code	p<char>
Description	
The prompt character will be displayed if preceded by the "\" character. <char>= Define prompt character and do not display <char>= \b: Display prompt character (default)	
Example	
Do not display LYNX prompt (prompt character is >). PRINT "\ep>"	

This command enables/disables the display of the controller prompt.

SET MASTER ADDRESS

Function	Set Master Address
Code	m<char>
Description	
<char>=A-Z, a-z, 0-9	
Example	
Set master address to Z. PRINT "\emZ"	

This command identifies to the HMI which LYNX controller product in a multidrop system is the master controller.

The character <char> will be the appellation of the master controller. When a number is entered on the keypad, this character is prefixed on the transmission.

If this setting is used, the HMI will have party mode enabled.

PARTY MODE ENABLE / DISABLE

This command will specify whether or not HMI is being used in a multidrop system receiving input from and outputing to several LYNX controller product nodes in a motion system.

In most situations the programmer will need to specify to the HMI the party mode address of the controller which is operating as the master using the "Set Master Address" command.

Usage Note! The HMI Party Mode Address is fixed as the uppercase character "H" and cannot be changed. Ensure that no other system LYNX controller products use this character as a node address.

Function	Party Mode Enable/Disable
Code	<flag>n
Description	
Must cycle power for the change to be active. <flag>= 0: Party Mode disabled (default) <flag>= 1: Party Mode enabled	
Example	
Enable Party Mode. PRINT "\e1n"	

SET HMI MODE

This command will set the HMI mode of operation. As detailed earlier in this document, these modes can be switched "on-the fly" inside a LYNX program.

This feature is extremely useful in instances where the user may want to input a register or thumbwheel value into a LYNX variable and then have the HMI function as a terminal display.

Usage Note! When switching the mode of operation "on-the-fly" it isn't necessary to write the mode setting to the NVM. Use the <flag>=0 parameter.

Function	Mode
Code	<mode#><flag>M
Description	
<mode#> = 2: Register Mode <mode#> = 1: Programmable Display Mode <mode#> = 0: Thumbwheel Mode <flag>= 1: Write mode to nonvolatile memory <flag>= 0: Do not write mode to nonvolatile memory	
Example	
Set thumbwheel mode and save mode setting. PRINT "\e01M"	

START-UP PROGRAM ENABLE / DISABLE

Function	Enable/Disable Start-up Program
Code	<flag>g
Description	
<flag>= 1: Run program <flag>= 0: Do not run program (default)	
Example	
Run program at power-up. PRINT "\e1g"	

This command specifies whether or not the HMI will run a start-up program located at address 0 of the HMI memory.

A start-up program would likely be a “Welcome Screen” or possibly initiate a homing routine in the motion system.

Usage Note! The start-up program will always reside at program location 0 of the HMI.

BACKLIGHT ON / OFF

Function	Backlight ON / OFF
Code	<flag>l
Description	
<flag> = 0: Backlight OFF (default) <flag> = 1: Backlight ON	
Example	
Turn the backlight ON. print "\e1l"	

This command specifies whether or not the HMI LCD display backlighting is on or off.

SET CONTRAST

Function	Set Contrast
Code	<num>z
Description	
Sets the contrast level of the LCD display <num>= 1 - 6: 1 (default)	
Example	
Set contrast level 3. PRINT "\e3z"	

This command specifies the contrast level of the LCD display of the HMI.

The contrast setting range is 1 - 6, with 1 being the default level.

FUNCTION KEY ENABLE / DISABLE

This command enables and disables the press and release operation of the HMI function key set F1 through F4.

The table below simplifies the configuration of the function keys.

Function	Function Key Enable / Disable
Code	<num>F
Description	
Enables / disables the press and release functions of the function keys F1 - F4. <num> = 0 - 255 (this setting will specify which individual function key press and release functions are enabled/disable) See note and table.	
Example	
Enable F1 Press and F2 Release. PRINT "\e18F"	

Usage Note! When enabling a function key, the "Set Function Key Address Pointer" command will be used to point the appropriate press/release function of the key to a location in HMI memory where the programmed instruction resides that will be performed upon the press/release of the function key.

HMI Function Key Table							
Enable / Disable Press Function				**Enable / Disable Release Function**			
F4	**F3**	**F2**	**F1**	**F4**	**F3**	**F2**	**F1**
128	64	32	16	8	4	2	1

Table A.2: HMI Function Key Setup

The above table shows the numbers which enable the press/release of the function keys. The number will be additive for the function keys you desire to use. For example: the press function used for all four function keys would be 128 + 64 +32 + 16 = 240, thus entering PRINT "\e240F" would enable the press function for F1 - F4.

ALTERNATE FUNCTION KEY ENABLE / DISABLE

Function	Alternate Function Key Enable / Disable
Code	<num>A
Description	
Enables / disables the press and release functions of the alternate function keys F5 - F6. <num> = (see table) (this setting will specify which individual function key press and release functions are enabled/disabled) See note and table.	
Example	
Enable F5 and F6 Press. PRINT "\e48A"	

This command enables and disables the press and release operation of the HMI alternate function key set F5 and F6.

The table below simplifies the configuration of the alternate function keys.

Usage Note! When enabling an alternate function key the, "Set Function Key Address Pointer" command will be used to point the appropriate press/release function of the key to a location in HMI memory where the programmed instruction resides that will be performed upon the press/release of the function key. See Appendix B: Function Keys for an example program.

HMI Alternate Function Key Table					
Enable / Disable Press Function			**Enable / Disable Release Function**		
	F6	**F5**		**F6**	**F5**
	32	16		2	1

Table A.3: HMI Alternate Function Key Setup

The above table shows the numbers which enable the press/release of the alternate function keys. The number will be additive for the alternate function keys you desire to use. For example: the press function used for both alternate function keys would be 32 + 16 = 48, thus entering PRINT "\e48A" would enable the press function for F5 - F6.

SET FUNCTION KEY PROGRAM ADDRESS POINTER

This command specifies the HMI memory address that the function key will point to. Upon the specified press or release of the key, the program at this address will execute.

This address will contain a string saved using the "Store Program" command. This string may be a LYNX instruction that is transmitted to the LYNX controller product to run an immediate mode command or LYNX label, execute a program, etc. It may also be an HMI command run locally.

Use of this is essential for an enabled function key to operate.

N **Usage Note!** The function key **MUST** point to an HMI memory location. This pointer command works hand-in-hand with the "Store Program" command. When the function key is pressed/ released it will execute the code string contained at the address specified by the pointer.

Function	Set Function Key Program Address Pointer
Code	<num>f<char>,<adx>\b
Description	
Points the press/release of a function key to an HMI memory location where resides the command, LYNX program, or process activated by the function key. <num> = 1: F1 <num> = 2: F2 <num> = 3: F3 <num> = 4: F4 <num> = 5: F5 <num> = 6: F6 <char> = P: Press Function <char> = R: Release Function <adx> = HMI Program Address (0 - 4095)	
Example	
Set the press of F1 to point to address 2000. PRINT "\e1fP,2000/b"	

User Storage Access Escape Codes

RUN PROGRAM

Function	Run Program
Code	<adx>G
Description	
This command will execute a string of program text at a specified location in HMI memory. <adx> = Address (0 - 4095)	
Examples	
Execute a program at location 1500. PRINT "\e1500G"	

This command will execute a string of text stored at a specified location in user storage space.

The string which this command will run will be previously stored using the "Store Program" command.

STORE PROGRAM

Function	Store Program
Code	<adx>P<char>,<string>\t
Description	
This command will store a string of program text at a specified location in HMI memory. <adx> = Address (0 - 4095) <char> = T: Transmit program <char> = L: Run program locally <string> = Program text string	
Examples	
TRANSMITTED PROGRAM Store a program at memory location 2000, string will execute a LYNX program labeled RUN_ONCE when run. PRINT "\e2000PT,RUN_ONCE\t" **LOCAL PROGRAM** Store a program at location 1500 that will turn on the backlight. PRINT "\e1500PL,\e1l\t"	

This command will store a string of program text at a specified location in user storage space.

The program text string may contain escape codes, register data, and text.

This text string can be either transmited to a remote location, such as a LYNX controller product to execute a program or run an immediate mode LYNX command, or it can be a string that is run locally in the HMI itself. An example of a locally run program would be in a case where you would use a function key to change an HMI setting.

This stored text string may be executed one of two ways, by a function key which has been pointed to this location by the "Set Function Key Address Pointer" command, or by using the "Run Program" command.

Usage Note! if the program text string contains register data, it must be embedded in the following fashion: \e<reg#>R, where <reg#> is equal to the register number (0-63). The reference to the register number will expand to the current register value.

Register Escape Codes

SET REGISTER VALUE

Function	Set Register Value
Code	<reg#>v<string>\b
Description	
This command will set the specified register to a value. <reg#> = Register (0 - 63) <string> = Register value (10 digits max. + sign)	
Examples	
Set register 15 to 2,000,000 PRINT "\e15v2000000\b"	

This command will store a string of up to 10 digits of numeric data to one of 64 registers in the HMI.

An example use for this data is to manipulate LYNX variables.

SET REGISTER SCREEN POSITION

This command will specify the location on the HMI LCD where the register will be displayed.

Up to four registers may be displayed on each screen. This command will be used in both register and thumbwheel mode.

Function	Set Register Screen Position
Code	<reg#>s<row>,<col>\b
Description	
This command will set the specified register to be displayed at a specified location on the HMI LCD. <reg#> = Register (0 - 63) <row> = Row Number (1 - 4) <col> = Column Number (1 - 20)	
Examples	
Set register 15 to display at row 2 column 1. PRINT "\e15s2,1\b"	

SET REGISTER PROGRAM ADDRESS

This command will specify an address in HMI memory where a program linked to a register is stored.

The mode setting specifies the register event which will take place upon the press of the "Enter" key, or the press or release of a programmed function key that affects the active register (see Set Register Transmit Method).

If the mode setting is in the range 0 to 4095, which are valid HMI memory addresses, the program stored at that address will be run when the register is entered.

|If the setting is 4096, the register value will be transmited. If 4097, the register value will be stored and no event will occur.

Function	Set Register Program Address
Code	<reg#>a<mode#>\b
Description	
This command will link the specified register to an address in HMI memory. <reg#> = Register (0 - 63) <mode#> = 0 - 4095: Run Program at the specified address <mode#> = 4096: Transmit register value <mode#> = 4097: Store register value only, no action performed	
Examples	
Link register 15 to program address 2000. PRINT "\e15a2000\b"	

SET REGISTER TRANSMIT METHOD

Function	Set Register Transmit Method
Code	<fkey#>x
Description	
This command will run the register program address when the defined function key is pressed. The cursor must be on the register displayed on the screen. <fkey#> = Function key (1 - 6) <fkey#> = 0: Runs register program address when the ENTER key is pressed. This allows a program to be attached to the entry of a register.	
Examples	
Run program when F2 is pressed. PRINT "\e2x"	

This command will set the transmit method of the active register.

If the ENTER key is used, <fkey#>=0, the register value will be saved and the program at the register program address will be run.

Usage Note! The cursor **MUST** be on the desired register on the display.

Usage Note! The release function of the function keys **MUST** be disabled if not used.

SET NO REGISTERS ACTIVE

Function	Set No Registers Active
Code	0u
Description	
This command will deactivate any active registers, will not clear the display.	
Examples	
Set no registers active. PRINT "\e0u"	

This command will deactivate any active registers.

SET REGISTER ACTIVE AND DISPLAY

This command will activate the specified register(s) and display them on the HMI screen in the positions specified by the "Set Register Screen Position" command.

Function	Set Register Active and Display
Code	<reg#>U
Description	
Must be in register mode for this command to function. <reg#> = Register Number (0 - 63)	
Examples	
Activate registers 7 and 13. PRINT "\e7U\e13U"	

Usage Note! The HMI **MUST** be in register mode for this command to function.

EXPAND REGISTER VALUE IN USER PROGRAM

This command will be embedded within the text string of the "Store Program" command.

Function	Expand Register Value in User Program
Code	<reg#>R
Description	
This command will expand the specified register value in a user program. <reg#> = Register Number (0 - 63)	
Examples	
Store a program at address 2000, transmit the program. Expand the value of register 15 as a parameter of the slew instruction. PRINT "\e2000PT,SLEW\e12R\t"	

GET THUMBWHEEL DATA

Function	Get Thumbwheel Data
Code	<tw#>W
Description	
This command will transmit the specified thumbwheel. <tw#> = Thumbwheel Number (1 - 4)	
Examples	
Send Thumbwheel 3 current value to the LYNX controller product. PRINT "\e3W"	

This command will transmit the current value of the specified thumbwheel to the LYNX controller product.

HMI must be in thumbwheel mode. PRINT "\e01M" for this to function.

CLEAR ALL THUMBWHEELS USED

Function	Clear All Thumbwheels Used
Code	0t
Description	
This command will reset the HMI to a "no thumbwheels in use" condition.	
Examples	
Clear thumbwheels used. PRINT "\e0t"	

This command will reset the HMI to a "no thumbwheels in use" state.

DISPLAY THUMBWHEEL VALUES

Function	Display Thumbwheels
Code	0T
Description	
This command will display the values of all thumbwheels in use.	
Examples	
Display thumbwheels. PRINT "\e0T"	

This command will display all thumbwheels currently in use.

WARRANTY

TWENTY-FOUR (24) MONTH LIMITED WARRANTY

Intelligent Motion Systems, Inc. (“IMS”), warrants only to the purchaser of the Product from IMS (the “Customer”) that the product purchased from IMS (the “Product”) will be free from defects in materials and workmanship under the normal use and service for which the Product was designed for a period of 24 months from the date of purchase of the Product by the Customer. Customer’s exclusive remedy under this Limited Warranty shall be the repair or replacement, at Company’s sole option, of the Product, or any part of the Product, determined by IMS to be defective. In order to exercise its warranty rights, Customer must notify Company in accordance with the instructions described under the heading “Obtaining Warranty Service.”

This Limited Warranty does not extend to any Product damaged by reason of alteration, accident, abuse, neglect or misuse or improper or inadequate handling; improper or inadequate wiring utilized or installed in connection with the Product; installation, operation or use of the Product not made in strict accordance with the specifications and written instructions provided by IMS; use of the Product for any purpose other than those for which it was designed; ordinary wear and tear; disasters or Acts of God; unauthorized attachments, alterations or modifications to the Product; the misuse or failure of any item or equipment connected to the Product not supplied by IMS; improper maintenance or repair of the Product; or any other reason or event not caused by IMS.

IMS HEREBY DISCLAIMS ALL OTHER WARRANTIES, WHETHER WRITTEN OR ORAL, EXPRESS OR IMPLIED BY LAW OR OTHERWISE, INCLUDING WITHOUT LIMITATION, **ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE**. CUSTOMER’S SOLE REMEDY FOR ANY DEFECTIVE PRODUCT WILL BE AS STATED ABOVE, AND IN NO EVENT WILL THE IMS BE LIABLE FOR INCIDENTAL, CONSEQUENTIAL, SPECIAL OR INDIRECT DAMAGES IN CONNECTION WITH THE PRODUCT.

This Limited Warranty shall be void if the Customer fails to comply with all of the terms set forth in this Limited Warranty. This Limited Warranty is the sole warranty offered by IMS with respect to the Product. IMS does not assume any other liability in connection with the sale of the Product. No representative of IMS is authorized to extend this Limited Warranty or to change it in any manner whatsoever. No warranty applies to any party other than the original Customer.

IMS and its directors, officers, employees, subsidiaries and affiliates shall not be liable for any damages arising from any loss of equipment, loss or distortion of data, loss of time, loss or destruction of software or other property, loss of production or profits, overhead costs, claims of third parties, labor or materials, penalties or liquidated damages or punitive damages, whatsoever, whether based upon breach of warranty, breach of contract, negligence, strict liability or any other legal theory, or other losses or expenses incurred by the Customer or any third party.

OBTAINING WARRANTY SERVICE

Warranty service may obtained by a distributor, if the Product was purchased from IMS by a distributor, or by the Customer directly from IMS, if the Product was purchased directly from IMS. Prior to returning the Product for service, a Returned Material Authorization (RMA) number must be obtained. Complete the form at http://www.imshome.com/rma.html after which an RMA Authorization Form with RMA number will then be faxed to you. Any questions, contact IMS Customer Service (860) 295-6102.

Include a copy of the RMA Authorization Form, contact name and address, and any additional notes regarding the Product failure with shipment. Return Product in its original packaging, or packaged so it is protected against electrostatic discharge or physical damage in transit. The RMA number MUST appear on the box or packing slip. Send Product to: Intelligent Motion Systems, Inc., 370 N. Main Street, Marlborough, CT 06447.

Customer shall prepay shipping changes for Products returned to IMS for warranty service and IMS shall pay for return of Products to Customer by ground transportation. However, Customer shall pay all shipping charges, duties and taxes for Products returned to IMS from outside the United States.

IMS™ INTELLIGENT MOTION SYSTEMS, INC.
Excellence in Motion™

P.O. Box 457, 370 N. Main Street
Marlborough, CT 06447 U.S.A.

Phone: 860/295-6102
Fax: 860/295-6107
Email: info@imshome.com
Home Page: www.imshome.com

TECHNICAL SUPPORT
Eastern U.S.
Phone: 860 / 295-6102
Fax: 860 / 295-6107
E-mail: etech@imshome.com
Western U.S.
Phone: 760 / 966-3162
Fax: 760 / 966-3165
E-mail: wtech@imshome.com

IMS MOTORS DIVISION
105 Copperwood Way, Suite H
Oceanside, CA 92054
Phone: 760 / 966-3162
Fax: 760 / 966-3165
E-mail: motors@imshome.com

IMS EUROPE GmbH
Hahnstrasse 10, VS-Schwenningen
Germany D-78054
Phone: +49 / 7720 / 94138-0
Fax: +49 / 7720 / 94138-2
Email: info@imseuropehome.com
European Sales Management
4 Quai Des Etroits
69005 Lyon, France
Phone: +33 / 4 7256 5113
Fax: +33 / 4 7838 1537
Email: bmartinez@imshome.com
German Sales/Technical Support
Phone: +49 / 35205 / 4587-8
Fax: +49 / 35205 / 4587-9
Email: hruhland@imshome.com