

Analog Input/Joystick Module

The Analog Input/Joystick Expansion Module adds two 0 to 5 volt analog input channels to the MicroLYNX System. Both channels can be used for data acquisition, or either channel can be used to directly control motion. This offers the user the capability of receiving input from a variety of analog sources such as temperature or pressure sensors, and then controlling events based upon those inputs.

The user-selected Joystick channel can be programmed to set the range, zero, deadband and sensitivity. Each channel uses a 12 bit D/A converter for better resolution as well as a fixed single pole analog filter with a cutoff frequency of 658 Hz to reduce the electrical noise that can be present in industrial environments.

The Analog Input/Joystick Module can be installed in any free slot, however only one (1) module can be used per MicroLYNX.



Electrical Specifications

Analog Input Voltage Range	0 to +5 Volts
Resolution	12 Bits
Offset	±2LSB
Integral Linearity Error	±2LSB
Differential Linearity Error	±3/4LSB
Absolute Maximum Voltage at Inputs	±24 Volts
Joystick Reference Voltage	+5 Volts
Precision Calibration Reference Voltage (±0.2%)	+4.096 Volts
Calibration Reference Voltage Tolerance	±2 %
Analog Input Filter Cutoff Frequency	658 Hz

Pin #	Connector Option	
	8 Position Phoenix	10 Pin Header
1	+5V (Joystick Reference)	+5V (Joystick Reference)
2	AIN 1	GND
3	GND	AIN 1
4	+5V (Joystick Reference)	+5V (Joystick Reference)
5	AIN 2	GND
6	GND	AIN 2
7	4.096V (Calib. Reference)	4.096V (Calib. Reference)
8	GND	GND
9		GND
10		N.C.

Table 11.7: Analog Input/Joystick Module Pin Configuration

Installing the Analog Input/Joystick Module

To install the Analog Input/Joystick Expansion Module in your MicroLYNX, perform the following in accordance with Figure 11.12.

- 1) Remove the two retaining screws (A) from the cover.
- 2) Remove the blank panel (1, 2, or 3) from the desired slot you want to use.
- 3) Carefully press the Expansion Module (B) into place by plugging the 28 pin connector into the desired receptacle (C, D, or E) and snapping it into place under the retaining clips (F).
- 4) Reinstall the MicroLYNX cover.
- 5) Affix the labels supplied with the Module as shown.

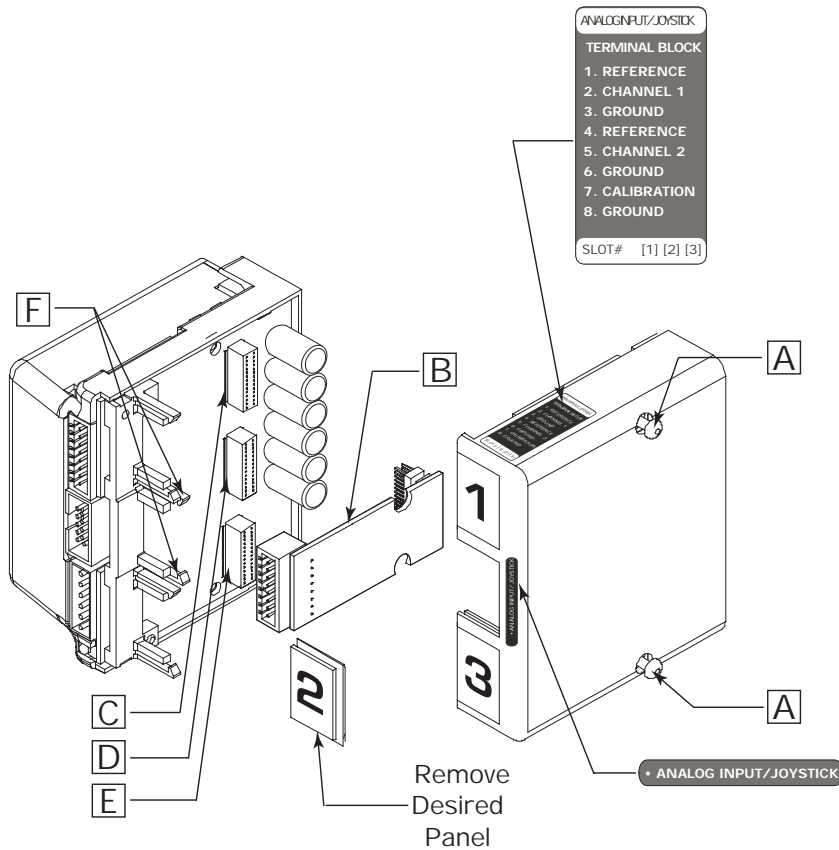


Figure 11.12: Installing the Analog Input/Joystick Module

Instructions & Variables Specific to the Analog Input/Joystick Module

There are several new enhancements to the MicroLYNX instruction set which add the functions of the Analog Input/Joystick Interface Module while maintaining backward compatibility with the modular LYNX System. The following instructions and variables are specific to the Analog Input/Joystick Interface Module. These are introduced here and covered in more detail in Part III, Software Reference.

Instruction	Usage	Description
IJSC	IJSC	CALIBRATE JOYSTICK INSTRUCTION: Supports the Analog/Joystick Interface Module when operating in joystick mode. Execution of this command followed by moving the connected joystick over its range of motion and back to center, and then pressing the "ENTER" key, or letting it time out for 30 seconds calibrates the joystick. This instruction allows for rapid calibration of the joystick.
Variable	Usage	Description
ADS	ADS<chan>=<aunit>,<func>,<law>	ANALOG INPUT SETUP VARIABLE: Supports the Analog Input/Joystick Module. <chan> = channel # (1 or 2) <aunit> = converts analog units to motor steps (velocity = aunit x munit) <mode> = 1 - analog, 2 - joystick <law> = Adjusts joystick position to motor velocity transformation. 1 = linear, 2 = square law, 3 = cube law
AIN	<var>=AIN<chan>	READ ANALOG INPUT CHANNEL: Causes a read of analog input channel <chan>, data is saved as variable <var>.
JSC	JSC=<num>	JOYSTICK CENTER POSITION VARIABLE, updated by IJSC command or directly as shown.
JSDB	JSDB=<num>	JOYSTICK DEADBAND VARIABLE, updated by ISJC command or directly as shown.
JSFS	JSFS=<num>	JOYSTICK FULL SCALE VARIABLE, updated by ISJC command or directly as shown.
Flag	Usage	Description
JSE	JSE=<flg>	JOYSTICK ENABLE FLAG: <flg> = 1 enables joystick function, <flg> = 0 (default) disables.

Table 11.8: Analog Input/Joystick Module Software Command Summary

Error Codes

In addition to the instructions and variables, the following error codes have been added to support the inclusion of the Analog Module and aid in troubleshooting the MicroLYNX System:

- 1201 Selected Analog Board not installed.
- 1202 Analog channel number not available.
- 1204 Analog option not installed.
- 1205 Analog VALUE out of range, possibly defective Board.

- 2101 Analog RANGE not allowed.
- 2102 Analog destination/source not allowed.
- 2103 Analog Destination/Source already used.
- 2104 Invalid Analog Channel number.
- 2105 Analog LAW not allowed.
- 2106 Can't enable Joystick while in motion, or can't exec motion cmd with Joystick enabled.

- 9014 Analog input not allowed for data.

The ADS Variable (A to D Setup)

The ADS variable is the heart of the MicroLYNX Analog Input/Joystick Interface Module. There are three parameters that control how the module will respond to input. It is used as follows:

ADS <chan>=<aunit>,<mode>,<law>

<chan>: Is the analog input channel that will be used, either 1 or 2.

<aunit>: This parameter sets the relationship between the analog input and units that are convenient to the user. In analog (User) mode the aunits parameter is the number of user units corresponding to the Analog Module full scale. In Joystick (Velocity) mode the aunits parameter is the number of munits/second corresponding to the Joystick Full Scale (JSFS) parameter.

<mode>: The mode parameter controls whether or not the input is used for velocity control; 1 = analog input, 2 = velocity or joystick mode.

<law>: Controls the sensitivity of the velocity with respect to the analog input. The effect of the analog input can be linear, square or cube. <law> applies to velocity mode only.

Here are two examples that illustrate the ADS variable:

Example 1

A pressure transducer is connected to input 1. The transducer output is 10 psi/volt. Vref represents the voltage at the Input to the Analog Joystick Module corresponding to full scale. Vref as measured at pin 1 on the Analog Joystick Module is 5.05 volts. Thus aunits for channel 1 is 10 psi/volt x 5.05 volts or 50.5. The value returned by an analog read of Channel 1 will be in psi. Note that the full scale output of the transducer does not have to equal the Analog Module full scale. This setup would be expressed thus:

ADS 1=50.5, 1

Example 2

A 1.8 degree (per full step) motor connected to a lead screw with a lead of .1 inches/rev. The step motor drive is set for 32 usteps per full step. A joystick is connected to channel 1. To program speed and motion in inches set munits to (32 pulses/1.8 degrees) x (360 degrees/1 rev) x (1 rev/.1 inches). If a maximum speed of 3 inches/second is desired while in Joystick operation set aunits for channel 1 to 3. For linear Joystick operation the setup command is ADS 1 = 3,2,1

Typical Functions of the Analog Input/Joystick Module

There are three program examples that will illustrate the use of the Analog Input/Joystick Module. In each case a 1kΩ potentiometer is used to emulate a sensor for analog input mode and a joystick for velocity mode.

Use the connection configuration shown in figure 11.13 below, a joystick or a sensor would be connected the same way.

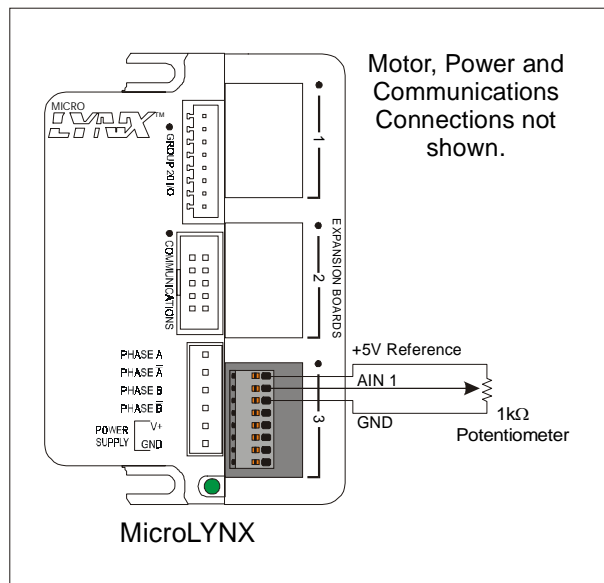


Figure 11.13: Analog Input Module Exercise Connection

Exercise 1: Velocity (Joystick) Mode

Here the potentiometer is emulating a joystick. Enter and execute the following program. When the voltage on AIN 1 is roughly 100mV either side of 2.5 volts it will be in the deadband range of the joystick. When less than 2.4 volts, the axis will accelerate in the minus (-) direction. When more than 2.6 volts, it will accelerate in the positive (+) direction. The velocity will increase as the voltage decreases from 2.4 to 0, or increases from 2.6 to 5.0. This can be watched with a multimeter. In this exercise both the axis velocity and position will display to the terminal screen.

```
'****Parameters****
MSEL=256
MRC=100
MAC=100
MUNIT=51200
JSDB=100      `Joystick deadband =100 aunits
VM =10000     `max velocity 10,000 munits/sec
ADS 1=1000,2,1 `chan. 1,aunits=1000, joystick, linear law
JSE = 1      `enable joystick functions

'****Program****

PGM 1
PRINT "\e[2J"
  LBL RUN
    PRINT "\e[1;1HInput Channel = " , AIN
    PRINT "Axis Velocity = " , VEL
    PRINT "Axis Position = " , POS
  BR RUN
END
PGM
```

Exercise 2: Sensor Input 1

Here we pretend the potentiometer is a pressure transducer and use it to display a pressure value to the screen.

```
ADS 1=50.5,1      `set ADS to aunit=50.5,analog input mode

PGM 200
LBL PRNTPSI      `name program "PRNTPSI"
PRINT "\e[2J"    `ansi esc. sequence to clear display
PRINT "Pressure = " , AIN 1 , " PSI"
BR PRNTPSI      `loop to program beginning
END
PGM
```

Exercise 3: Sensor Input II

Once again our potentiometer is pretending to be a sensor. In this exercise the program will call up a subroutine based upon the voltage seen on AIN 1 and position the axis at an absolute position. The best analog to this example might be a flow control application.

```

    '****Parameters****
MUNIT=51200           `munits=51200
MAC=75               `acceleration current to 75%
MRC=50               `run current to 50%
ADS 1=5,1           `aunits 5, analog input mode
VAR LIMIT=0         `declare user var "LIMIT"

    '****Program****
PGM 200
LBL AINTST           `name program "AINTST"
LIMIT = AIN 1       `set user var "LIMIT" = AIN 1
CALL ATEST, LIMIT>3.5 `call ATEST if LIMIT is greater than 3.5 aunits
CALL BTEST, LIMIT<3.5 `call BTEST if LIMIT is less than 3.5 aunits
BR 200              `loop to beginning of program
END

    '****Subroutines****
LBL ATEST           `declare subroutine "ATEST"
VM=20               `max. velocity = 20 munits/sec.
MOVA 10             `index to abs. pos. 10
HOLD 2              `suspend prog. until motion completes
RET                 `return from subroutine

LBL BTEST           `declare subroutine "BTEST"
VM=5                `max velocity = 5 munits/sec.
MOVA 22             `index to abs. pos. 22
HOLD 2              `suspend prog. until motion completes
RET                 `return from subroutine
```