

# Byte Arts USB PortBrain Interface Board

Revision: 13.9.26 for Firmware 3.0E



## CONTACT AND ORDERING INFO:

Got to [www.bytearts.com/products/portbrain](http://www.bytearts.com/products/portbrain) for the latest information.

## DESCRIPTION:

The PortBrain I/O Board is a compact device which provides flexible, intelligent I/O capabilities to any computer and can even operate as a stand-alone device.

### Main features:

- **24 bits digital I/O.** Each pin is independently programmable. TTL-compatible.
- **4 analog inputs.** Each input accepts a 0-5V signal and produces a 10 bit result.
- **Programmable logic functions** – the PortBrain can be automatically control output bits depending on the state of inputs. It can continually scan the inputs and change any of the outputs according to a set of user-customizable logic conditions.
- **Built-in temperature sensor.** Easily monitor a temperature. Outputs can be programmed to automatically change based on temperature reading. Also accepts external thermistor.
- **Full-Speed USB interface.** Acts as a virtual COM (serial) port, so PC can send and receive data using standard COM port functions and a simple command set.
- **Compatible with Windows, Linux and OSX.**
- **Stand-alone operation.** Once programed, the PortBrain can operate without a PC.
- **EEPROM storage.** All settings are saved and recalled on power-up.
- **USB or external power.** Requires 5VDC from USB or an external power supply.
- **Programmable flash memory.** Firmware can be updated in the field. Custom firmware also available.
- **Includes Windows software** with easy-to-use graphical interface for configuring and monitoring the PortBrain.
- **Dynamic Link Library (DLL)** for Windows and Linux also available.

## INSTALLATION

The PortBrain uses a standard USB device interface class (the Communication Device Class) and so doesn't require a custom driver and should be automatically recognized by most operating systems. Some versions of Windows require a special INF file in order for the operating system to recognize the PortBrain. This file and installation instructions are available at [www.bytearts.com/products/portbrain](http://www.bytearts.com/products/portbrain).

## ELECTRICAL SPECIFICATIONS:

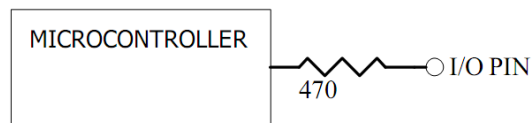
Power: 5VDC, 250mA max. Jumper-selectable USB power, or external power source.

Max Voltage on an I/O or Analog Input pin: 5.3V

Max Current sourced or sunk by an I/O pin: 25mA (Total of all I/O pins cannot exceed 200mA).

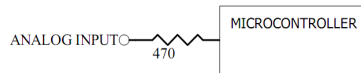
### I/O Pin Electrical Diagram:

Each I/O pin is a TTL input or output and has a 470 Ohm current-limiting resistor in series, as shown here:



### Analog Input Electrical Diagram:

Each analog input includes a 470 Ohm protection resistor, as shown here:

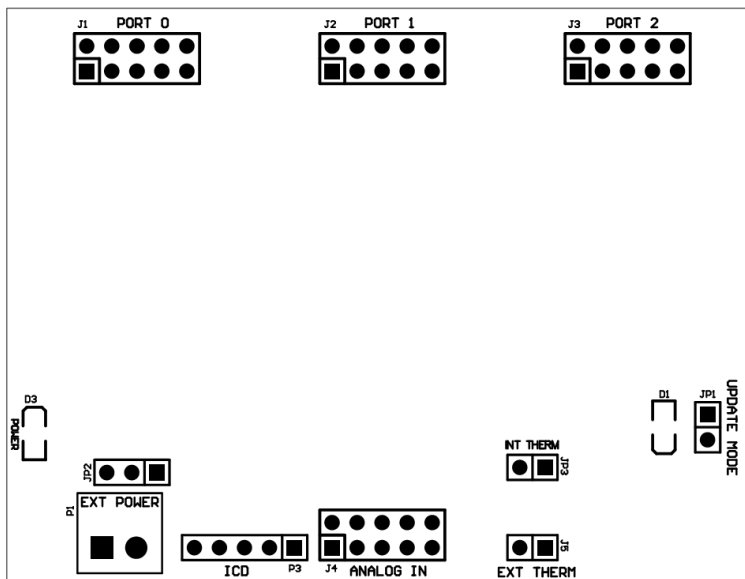


The maximum recommended source impedance for an analog input is 2.5 kOhms.

In addition, PORT 0 and PORT 2 have internal pull-up resistors which can be programmed to pull all inputs high (to 5V). The PortBrain defaults to having these pull-ups disabled – they can be enabled by sending the appropriate commands to the PortBrain.

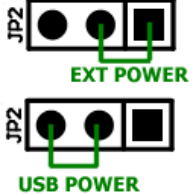
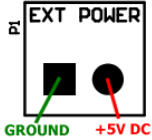
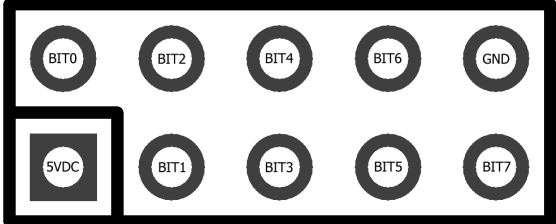
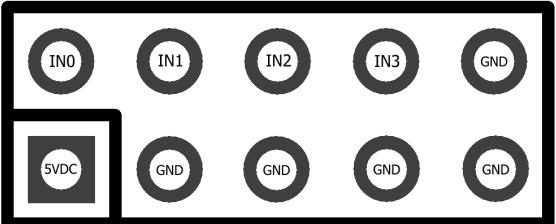
## CONNECTOR AND JUMPERS:

This diagram shows the location of each connector and jumper on the board:



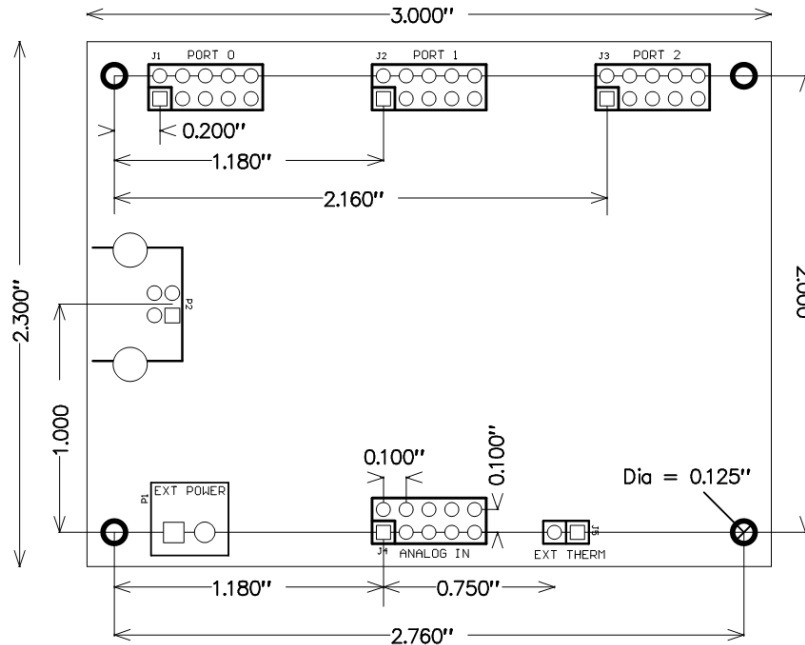
All connections are described in detail in the table below.

## Connectors, Jumpers and LED Pinouts and Description

Label	Description
JP1	When the PortBrain is powered up with the two pins on this jumper shorted, it will go into “bootload” mode which allows you to load new firmware. See “Updating the Firmware“ for more information. Note: there is also a command which can be sent to put the PortBrain into bootload mode without needing to change the jumper.
JP2	<p>Power select. Connect pins 1 and 2 together to use an external 5V power source (connect power source to connector P1). Connect pins 2 and 3 together to power the PortBrain from the USB connector.</p> 
JP3	Short the two pins together when using the on-board thermistor. Disconnect these pins when using an external thermistor.
JP5	If you want to use an external thermistor connect a 2.2K NTC thermistor to this input. Disconnect the pins of JP3 to disconnect the on-board thermistor.
D1	USB status indicator. Stays on continuously when USB connection is active. Blinks fast (4Hz) when USB is in standby mode. Blinks very slow (once every 3 seconds) when USB is not connected.
D3	Power indicator – on whenever power is connected to the PortBrain.
P1	<p>External Power Connection. If not using power from the USB connector, then connect external power source as shown here:</p> 
J1, J2, J3	<p>I/O Ports (PORT0, PORT1, PORT 2). Each port has 8 programmable I/O bits, plus power and ground.</p> 
J4	<p>Analog Inputs. There are 4 analog inputs on this connector. You can connect a 0-5V signal to each input.</p> 

## MECHANICAL LAYOUT:

This diagram shows the location of all the mounting holes and connectors. All dimensions are in inches. All the port pins are on a 0.1" grid so you can easily connect to a "proto board".



## COMMUNICATION PROTOCOL AND COMMAND SET:

The following information is only needed if you are writing your own software to control the PortBrain. If you are using the Windows software supplied with the PortBrain, you can skip to Using the Windows Software. Commands are sent to the PortBrain from the host computer over the USB connection. The PortBrain acts as a "virtual COM" port, and so looks like a normal RS232 (serial) port on the host. The software on the host system just needs to open the port (using any baud rate) and send out the ASCII characters for a command and then wait for a response back from the PortBrain.

### Command Protocol:

All commands and parameters are ASCII characters, all values are ASCII decimal (base ten) numbers. Each command must be terminated with a carriage return (ASCII value 13) or a line feed (ASCII value 10). If a command has any parameters, then there must be a space between the command and the parameter.

Once the PortBrain receives a command, it processes it immediately and returns a response. All responses are ASCII characters terminated with a carriage return (ASCII 13).

If a command is not recognized, or there is an invalid or missing parameter, then the PortBrain will respond with a ? (question mark) followed by an error code.

?1 - command is not recognized. All commands must be in upper case.

?2 – bad parameter value. A parameter value is invalid (not within the allowed range).

?3 - command is missing a parameter. Some commands require parameter(s) – there must be a space between the command and any parameters.

### Command Set

Command	Description	Return Value
	Analog Input Commands	
ADC<n>	Read Analog Input <n>, where n = 0..4. Input 4 is the built-in temperature sensor. Example: ADC3 – reads A/D input #3.	Returns a value between 0 to 1023 (inclusive). The value is proportional to the voltage on the specified input, where 0 = 0 Volts, and 1023 = Power Supply Voltage (nominally 4.8V).
TEMP	Reads the temperature sensor.	Returns the current temperature in Degrees Fahrenheit.
	Digital I/O Port Commands	
DIRRD<p>	Reads port direction register, where <p> = port number (0..2). Each port has 8 pins, each of which can be configured as an input or an output. Example: DIRRD1 – reads port direction register for PORT1.	Returns a value between 0 and 255 (8 bits), where each bit position corresponds to a pin of the specified port. If a bit is “1”, then the corresponding pin is an Input, if the bit is a “0”, the pin is configured as an output.
DIRWR<p> <v>	Writes to the port direction register, where <p> = port number (0..2), and <v> = an 8-bit value (0-255) where each bit position corresponds to a pin of the specified port. Only actual physical ports 0, 1 and 2 can be configured as inputs, virtual ports 3, 4 and 5 are always configured as outputs. Examples: DIRWR0 0 – sets up all PORT0 pins as outputs. DIRWR2 255 – sets up all PORT2 pins as inputs.	Returns “OK”.
PRTRD<p>	Reads all 8 pins of a port, where <p> = port number (0..5). Ports 0,1, and 2 are “real” physical ports, while ports 3,4 and 5 are “virtual” ports which aren't actually connected to the outside world and can be used to save values, test logic, etc..	Returns a value between 0 and 255 (8 bits), where each bit position corresponds to a pin of the specified port.

<b>Command</b>	<b>Description</b>	<b>Return Value</b>
PRTWR<p> <v>	Writes to all 8 pins of a port, where <p> = port number (0..5) and <v> = value (0..255). If any pins of the specified port are not configured as outputs, this command has no effect on those pins (only pins configured as outputs are affected).	Returns “OK”.
PU<p> <n>	Enable/disable pull-up resistors on Port <p>, where <p> = 0 or 2. If <n> = 0, the pull-ups are disabled. Note: pull-up resistors only have an effect on pins which are configured as inputs. Examples: PU0 1 – enables Port 0 pull ups. PU2 0 – disables Port 2 pull ups.	Returns “OK”.
PU<p>?	Query status of pull-ups on Port <p>, where <p> = 0 or 2.	Returns 0 or 1, where 1 = pullups enabled..
<b>Digital I/O Pin Configuration Commands</b>		
PINRD<p>	Reads the current state of pin <p>, where p = 0..31.	Returns 0 or 1. A “1” indicates that the pin is at a logic-high level (voltage is > 2V).
PINWR<p> <v>	Write a value to a pin <p> where p = pin number (0..31) and <v> = value (0 or 1). The pin must be configured as an output, otherwise this command has no effect. Example: PINWR5 1 – writes a 1 to pin 5.	Returns “OK”.
<b>Board Configuration Commands</b>		
BTLDR	Resets the PortBrain and restarts it in “bootload” mode where it is ready to accept a firmware update. This is equivalent to shorting the pins of JP1 on the board and unplugging and plugging the USB back in.	No return value is sent, and the COM port to which the PortBrain is assigned will no longer be available until the PortBrain is reconnected.
RECALL	Recall all settings from EEPROM.	Returns “OK”.

Command	Description	Return Value
SAVE	Save all current settings in EEPROM. Saves all the PLC program code, port configurations and the pull-up status to non-volatile memory. All settings are automatically recalled from this memory when the PortBrain is powered up.	Returns “OK”.
VER	Get firmware version number.	Returns the firmware version number as a string, such as “1.01”.

## UPDATING THE FIRMWARE

From time to time firmware updates are made available in order to add new features or fix bugs. Your PortBrain is shipped with the latest firmware already installed and so you don't normally need to install an update. You can check for information on the latest firmware and software versions at [www.bytearts.com/products/portbrain](http://www.bytearts.com/products/portbrain).

### Firmware Update Procedure

The following instructions show how to update or load a different version of the firmware on the PortBrain using a Windows PC. Instructions for updating the firmware using other operating systems are available on at [www.bytearts.com/products/portbrain](http://www.bytearts.com/products/portbrain).

NOTE: The firmware update procedure will erase any stored settings and set all I/O pins to inputs.

#### **Step 1. Put the PortBrain in Update Mode**

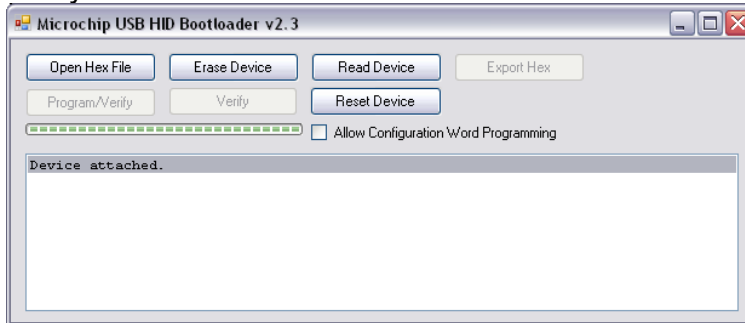
Method A: Connect the PortBrain to the PC, then run the PortBrain software. Once connected, go to the Options menu, then select “Update Firmware..” and the PortBrain will be rebooted in update mode. Proceed to Step 2.

Method B: Connect the pins of JP1 together, and then plug the PortBrain into a USB port on the host computer. The Windows “New Hardware Found” wizard may pop-up – simply select the defaults (press the OK button) at each step and Windows will install the required software (no special drivers are needed).



## Step 2. Run the HID Bootloader

1. Run the "updater program". If you have installed the PortBrain Windows software, then this program is installed already (Go to the Windows Start menu, and look under "All Programs->Byte Arts->PortBrain->Firmware Updater (HID Version)"), otherwise you will need to download it from [www.bytearts.com/products/portbrain](http://www.bytearts.com/products/portbrain). Run this program and you'll see a window like this:



Click on the "Open Hex File" button and select the firmware file (.HEX) to load. Firmware files are installed in the firmware folder (C:\Program Files\Byte Arts\PortBrain\Firmware) when you install the Windows software, and are also available for download at [www.bytearts.com/products/portbrain](http://www.bytearts.com/products/portbrain).

2. Next, click on the "Program/Verify" button and wait for the magic to happen (about 10-20 seconds).

## Reset the PortBrain

If you used Method B in Step 1, then remove the jumper from JP1 and then reset the PortBrain (unplug the USB). If you used Method A, then just unplug the PortBrain and then plug it back in.

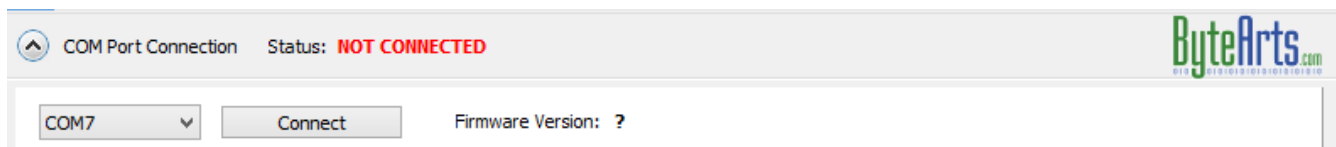
When you plug it back in, it will start up with the new firmware installed.

## USING THE WINDOWS SOFTWARE

The software can be downloaded from [www.bytearts.com/products/portbrain](http://www.bytearts.com/products/portbrain).

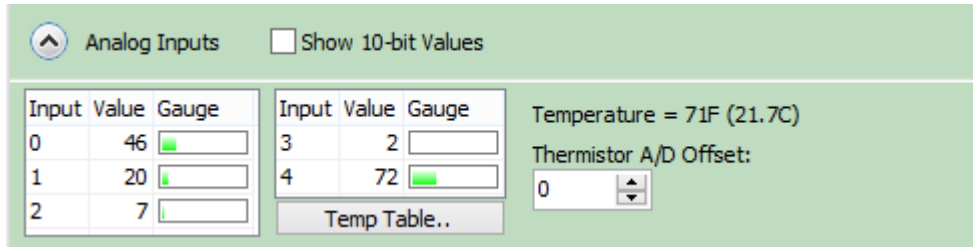
The main window is separated into 4 sections, each section can be collapsed or expanded by clicking on the up/down arrow button in the upper-left portion of the section.

### Section 1: COM Port Connection



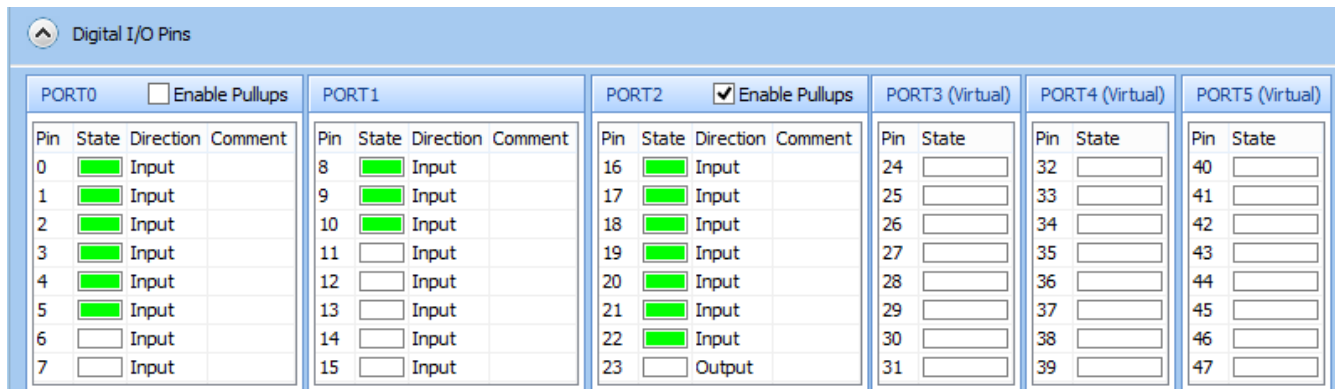
To have the software connect to the PortBrain, you must first select the COM port from the drop-down list and then press the Connect button. Without having a PortBrain connected you can still use the software to edit configurations and PLC programs, but of course will not be able to actually control anything or save settings in the EEPROM on the PortBrain.

## Section 2: Analog Inputs



This section is used to monitor the analog input values. There are five analog inputs, each of which measures a voltage from 0 to 5V, and displays the result value as either an 8-bit value (0 to 255), or a 10-bit value (0-1023). Normally, a thermistor is connected to the last input (number 4), and the software displays the current temperature reading from that thermistor using a built-in look-up table. An offset can be added to the thermistor reading to calibrate the temperature if needed.

## Section 3: Digital I/O Pins



This panel lets you view the status of all the input and output pins on the PortBrain. There are three 8-bit ports (Port0, Port1 and Port2) which are connected to header pins on the PortBrain and which can be connected to external devices such as switches, lights etc., but there are also three 8-bit “virtual” ports which are not connected to anything, but can be used internally as counters, variables or storage locations in PLC programs.

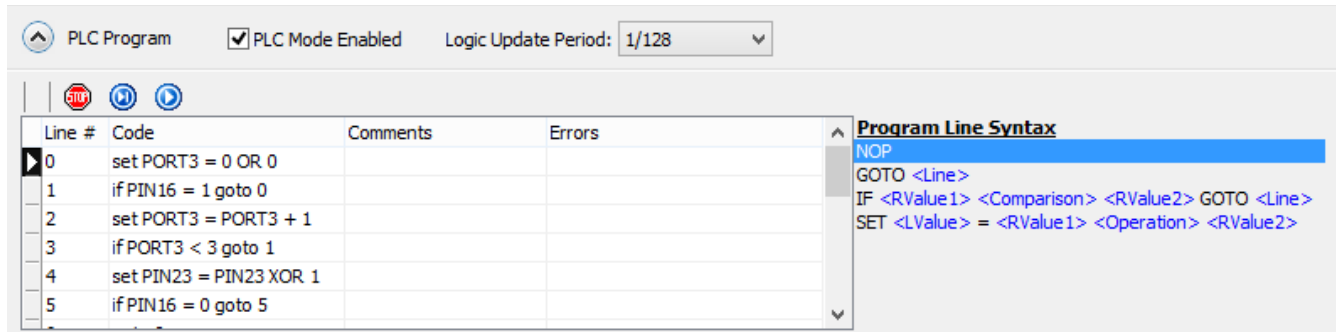
The “State” column shows the current digital signal level on the pin. A green box means the pin is set “high”, while if the box is white, then the pin is “low”. All pins are TTL-level outputs (0 – 5V).

You can click in the “Direction” column to change any pin to be either an input or an output. If a pin is set to be an output, then you can click in the “State” column to set the pin high (5V) or low (0V).

Port0 and Port2 have internally connected pull-up resistors which can be enabled in the software. This only affects pins which are configured as inputs, and when enabled, the pins will be pulled up to a logic high level (5V) if left unconnected. This is useful when a pin is connected to a “normally open” switch which would normally leave the pin “floating” (not connected to a voltage source or to a ground) when the switch is not closed.

Virtual port pins always act as outputs and so you can change their state anytime.

## Section 4: PLC Programming



The PortBrain can act as a stand-alone controller and as long as it is supplied with power, it will continually run any loaded PLC program. PLC programs are created by entering each instruction on a line – each line is error-checked as you enter it, so it is impossible to enter an invalid program. (Whether the program actually does what you meant it to do is another matter).

Before you can edit or run PLC programs, you must check the “PLC Mode Enabled” checkbox. The “Logic Update Period” can be used to adjust the speed at which each line of the PLC program is executed – running at a slower rate allows you to better watch the execution of the program and can be useful for implementing long delays.

There are only four instructions: NOP, GOTO, IF, and SET, but using these four instructions it is possible to create very sophisticated programs.

### PLC Instructions

**NOP** – this is a “do nothing” operation (literally it means “no operation”). It is mainly used to put in a delay, or as part of a loop where you don't want to do anything.

**GOTO <Line Number>** – this jumps execution to the specified line number.

Example: GOTO 5

**IF <RValue1> <Comparison> <RValue2> GOTO <Line Number>** - this instruction is used to check for a specified condition and then jump (GOTO) to a program line if the condition is TRUE. If the condition is NOT met (the condition is FALSE), then the program continues on to the next line.

It can check the state of an individual pin (either real or virtual), or the state of a port (all 8 bits), or an analog input. The variables <RValue1>, <RValue2> can be any of the following:

RValue Variables	Description
PORT0..PORT5	Yields the 8-bit value (0-255) of the specified port.
PIN0..PIN47	Yields the value (0 or 1) of the specified pin.

AIN0..AIN4	Yields the 8-bit value (0-255) of the specified analog input.
Constant value	Any number between 0 and 255.

The following types of comparisons can be done:

Comparison	Description
=	Result is TRUE if RValue1 equals RValue2.
<	Result is TRUE if RValue1 is less than RValue2.
>	Result is TRUE if RValue1 is greater than RValue2.
<>	Result is TRUE if RValue1 does NOT equal RValue2.

Examples:

IF PIN1 > 0 GOTO 7 - if PIN1 is set (not at a logic low), then goto line 7.

IF PORT0 = 255 GOTO 1 – if ALL the pins in Port0 are high, then goto line 1. (255 = all 8 bits set).

IF AIN0 > 20 GOTO 5 – if analog input 0 is greater than 20, then goto line 5.

**SET <Lvalue> = <RValue1> <Operation> <RValue2>** - this instruction is used to change the state of a pin or port, by performing an operation on up to two other values.

The variables <RValue1> and <RValue2> are the same as the IF instruction.

The <LValue> variable represents the output that will be affected, and can be an individual pin, or an 8-bit port.

LValue Variables	Description
PIN0..PIN47	A single pin. If the result of the operation is not 0, then the pin will be set high. The pin must be configured as an output for this to have any effect.
PORT0..PORT5	An 8-bit port. The result is assigned to all pins of the specified port. The least-significant bit of the value corresponds to the first pin (lowest numbered) of the specified port.

Operations which can be specified are:

Operation	Description
+	Adds RValue1 and RValue2.

-	Subtracts RValue2 from RValue1.
AND	Performs a bit-wise logic AND operation.
OR	Performs a bit-wise logic OR operation.
XOR	Performs a bit-wise logic Exclusive-OR operation.

Examples:

SET PIN0 = 0 + 0 - sets PIN0 low.

SET PORT1 = PORT1 + 1 – increments the current value of PORT1

SET PIN0 = PORT1 AND 7 – sets PIN0 high if any of the lower 3 bits of PORT1 are high.

**Running a PLC Program**

To enable running a PLC program, you must first check the “PLC Mode Enabled” checkbox. This enables the PLC programming toolbar. There are three buttons on the toolbar:

STOP – stops the running program.

STEP – executes the currently selected line of the program.

RUN – runs the program. Execution will continue until the STOP button is pressed. You can adjust the execution speed by changing the “Logic Update Period” setting.

While the program is running (or being stepped), the values of all the ports and analog inputs will be continually updated on the screen, so you can watch the results of the program. The STEP function is particularly useful to debug a program since it lets you watch each line get execute one at a time, and examine the results after each step.

**Running in Stand-alone Mode**

If you are going to run the program in stand-alone mode (meaning with no PC attached), then be sure that the PLC Mode Enabled checkbox is checked, and that you have saved the program and all the settings in the EEPROM on the PortBrain by pressing the “Save Settings in EEPROM..” button. After you have done this, when the PortBrain is powered up, it will automatically start executing the saved program.

**Saving and Recalling Settings**

You can save and recall all the current settings and PLC program to/from a file from the File->Save or File-Recall menus. Recalling a file will recall both the port settings (the state and direction of each pin) and the PLC program. When a file is recalled, it is loaded into the software, and if a PortBrain is connected the settings are transferred to the PortBrain, but are NOT automatically saved in the EEPROM (permanent memory) until you press the “Save Settings in EEPROM” button.