Affordable Comprehensive Transparrent
PERSONAL MECHATRONICS LAB

Personal Mechatronics Lab Microcontroller Board User Manual

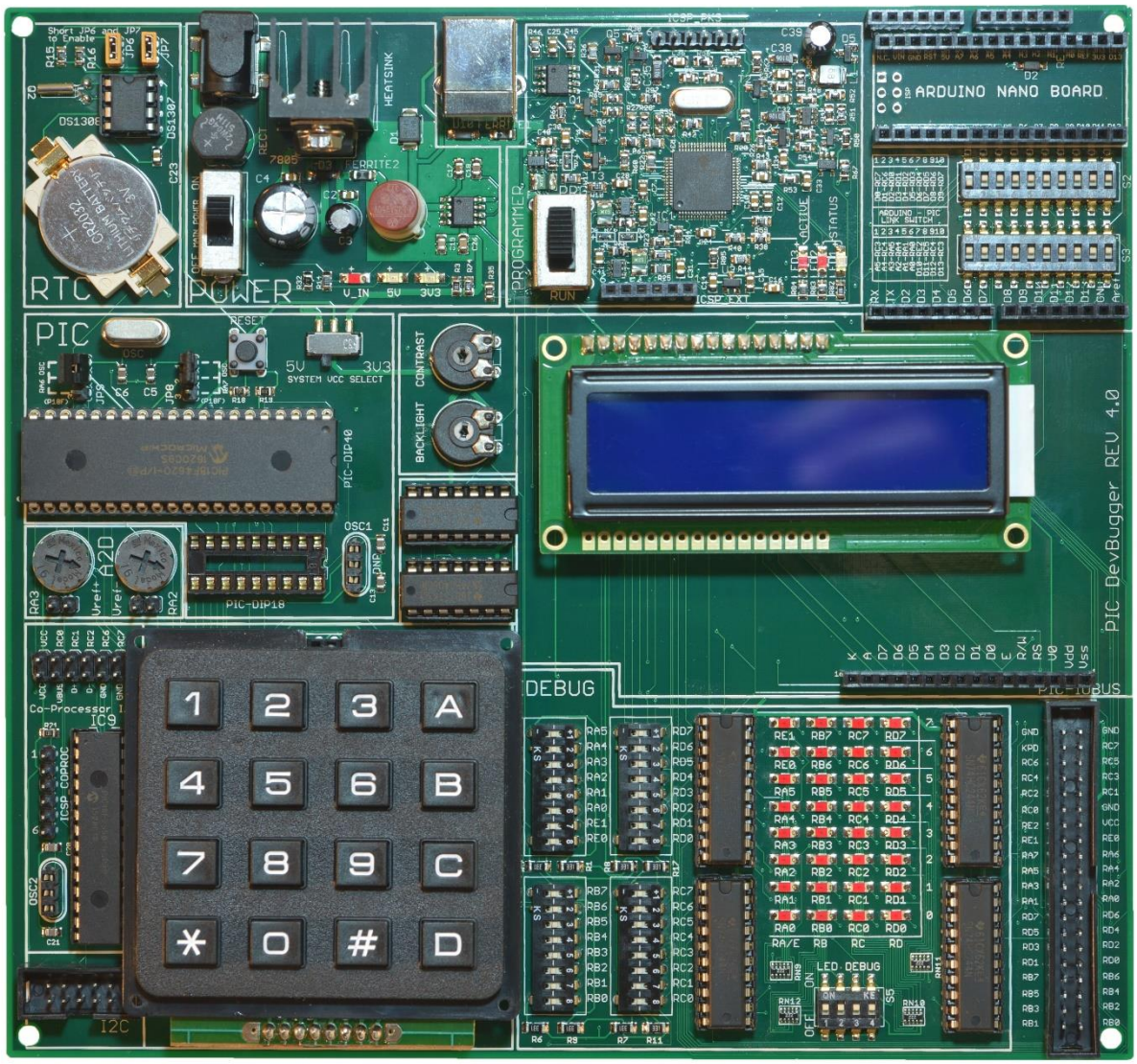

Table of Contents

1. Introduction

1.1 Overview

The *PIC Dev Bugger* development board was designed as a platform for development of mobile robotic control solution. Modules of the board includes: a full-speed USB2.0 programmer, 4x4 matrix keypad input, up to 4x20 character LCD, Real Time Clock, Analog to digital conversion, Arduino slave interface, and I2C serial port. One especially useful feature of this development board is the debug module, which monitor states of all user accessible pins of the onboard PIC microcontroller. Additionally, the user can pull up or down each pin connected to the debug module directly from the board, which allows quick and effective code debugging.

This board is especially designed to suit the needs of students learning to develop for embedded systems. The form factor and features of the board allows it to be used both as an exclusive code development platform or in system as the main processing and control system in a final design.

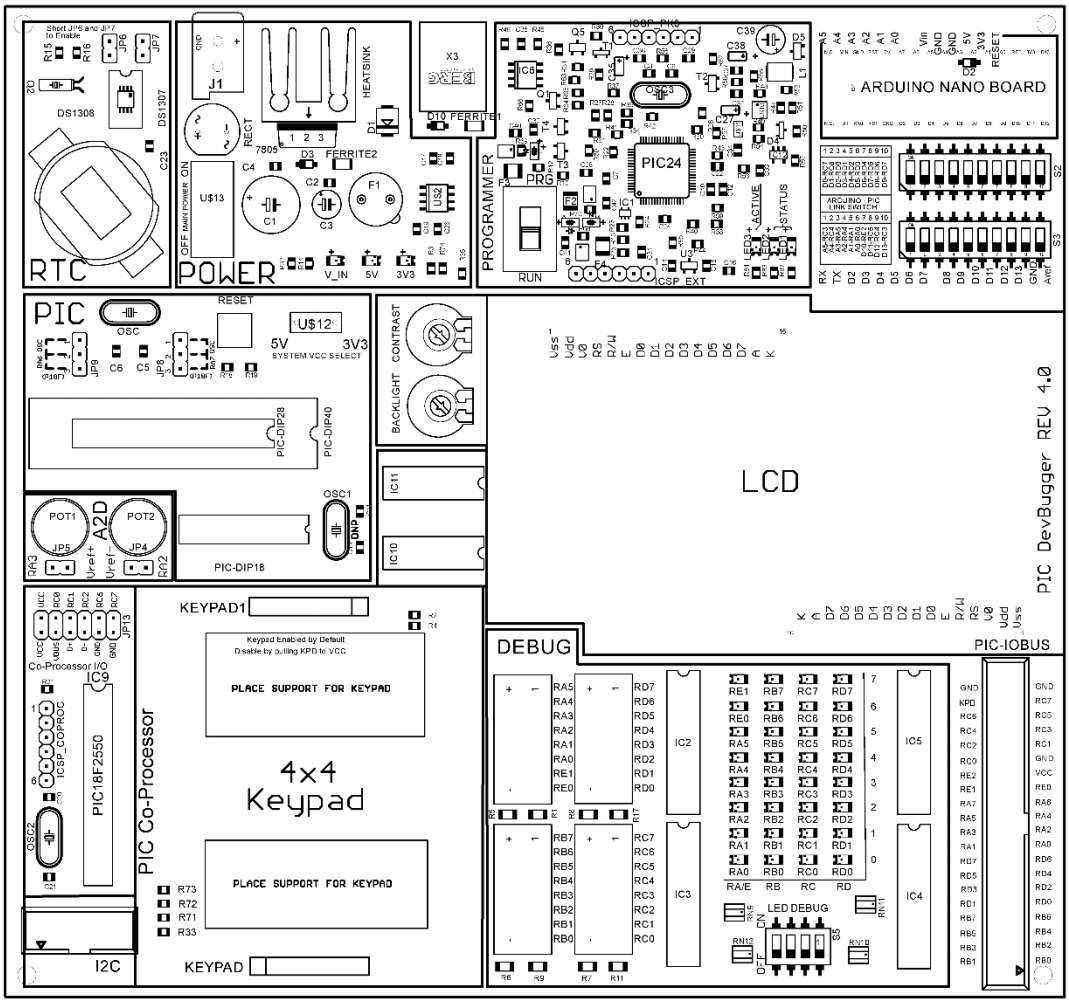

Figure 1: The PIC Microcontroller board

1.2 Features

- Open, modular, and simple design for learning purposes
- Supports 18-, 28-, and 40-pin packages for PIC16 and 18 families
- In-circuit PICkit™ 3 compatible High Voltage Programmer (works with MPLABX® IDE)
- Dual power source, USB or DC power supply input (from 7.5VDC to 17VDC)
- Dual system voltage support (5V and 3.3V system VCC)
- Debugging Module with 32 indicator LEDs and signal-emulation switches
- Keypad encoder chip is a full featured PIC, whose firmware can be modified for extra memory, parallel processing, and/or I/O pin extension for the primary MCU
- Real Time Clock peripheral with 32.768khz crystal and battery socket (RTC chip included)
- On-board socket for Hitachi HD44780 compatible LCD with contrast and backlight controls, support size up to 4x20 characters
- On-board 4x4 matrix keypad socket and PIC based encoder
- 40-pin I/O bus with ribbon cable connector
- Interchangeable oscillator clock (10MHz crystal included), with dedicated oscillator socket for 18-pin PIC.
- I^2C bus expansion socket
- On-board adjustable ADC voltage reference
- Arduino Nano board interface, with individually selectable pin pass-through to PIC.
- Arduino shield connector, with pin pass-through from both PIC and Arduino.

1.3 Included in the Box

The development kit come included with the following items:

- PIC Microcontroller development board
- CD/URL from which to download necessary software, drivers, and sample codes
- USB A to B cable
- 40-pin I/O bus cable
- HD44780-controlled LCD display (2x16 characters)
- 4x4 matrix keypad
- AC-DC adaptor unit (120V AC to 12V DC,.1A)

2. Operation

2.1 Operational Modes

The PIC *Microcontroller board* has two modes of operation, as follows:

Programming: Used to load compiled HEX code onto the main PIC device, as well as debugging of code on device. To enter this mode, flip the PROG/RUN slide switch of the programmer module to the PRG position. In this mode, the VPP, PGC and PGD pins of the main PIC device are disconnected from the main I/O bus, and connected instead to the programmer module. A PC application compatible with the PICkit™ 3 programmer, such as MPLABX® IDE, can be used to control loading and debugging in this mode.

Execution: This is the primary operational mode of the board. To enter this mode, flip the PROG/RUN slide switch of the programmer to the RUN position. In this mode, All I/O pins of the main PIC device are connected to the I/O bus, and code executes freely. The onboard programmer can be used independently as a PICkit™ 3 programmer for programming of external PIC boards or the onboard PIC keypad encoder. The ICSP_EXT headers can be used for this purpose when the development board is in this mode.

2.2 Connecting to the PC for Programming

To load HEX code to the PIC device, the *Microcontroller board* must connect to a PC:

1. Install the **MPLABX IDE** and/or **MPLABX IPE** distributed by Microchip
    a. The development board will be recognized as a PICkit™ 3 programmer
2. Connect the *Microcontroller* board to the PC using the included USB cable.
3. Power the board and turn it on. The board can be powered by the USB cable. However, use of the included wall adaptor in conjunction is strongly recommended.
4. Set the board to **Programming** mode. (Section 2.1)
5. Flip the power switch to the ON position. (Section 3.1)
6. The PC should detect the Microcontroller and install the driver automatically.
7. The PIC device on the board is now ready to be programmed. (Section **Error! eference source not found.**)

2.3 Customizing Board Operation

The *Microcontroller board* was designed with versatility in mind. To customize the operation of the board, several configuration jumpers and switches have been included, which must be set by the user. Before using the board for the first time, please ensure that the jumpers for each module have been configured as desired. More information about jumper settings for individual modules can be found in Section 3.

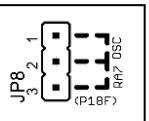

Figure 2: Jumper

For those unfamiliar with jumpers, a jumper is a set of 2 or more exposed pins that can be connected adjacently in pairs, using a 'shunt'. When two pins of the jumper are connected to each other, they are 'shorted'. As an example, if pins 1 and 2 in the figure are connected using a shunt, then we say we have 'shorted' pins 2+3, and we have configured the board for use with a P18F type microcontroller as labeled above the jumper.

2.4 Interfacing with External Circuits

While the board is operating in **Execution** mode, all I/O pins of the PIC microcontroller are directly connected to the main I/O bus socket. Using the provided ribbon cable, this bus can be used to interface directly with external circuitry. The I/O bus can provide some power at the working voltage of the board (selectable between 5V and 3.3V) and ground reference in addition to direct access to the pins of the main PIC. No high-power devices or circuitry (such as motors, fans, solenoids, or high powered lights) shall be powered from the VCC output on the development board. Also, note that the Keypad peripheral adds an extra pin to the bus (KPD pin), for on-the-fly enable/disable control, as described in Section 3.5. A detailed description of the pin connections on the ribbon cable is given in Section 3.8.

WARNING: the VCC supply on the bus is NOT recommended as an alternate method of powering the Microcontroller board. Especially when the board is operating in 3.3V system voltage mode. Power should always be supplied through the DC input jack when the board is used as part of an embedded system (voltage of 7.5 to 12 volts are recommended). The onboard power supply module will ensure delivery of clean DC power to all aspects of the development board.

NO guarantee is made for the continued integrity of the board in cases of such usage.

3. Board Modules

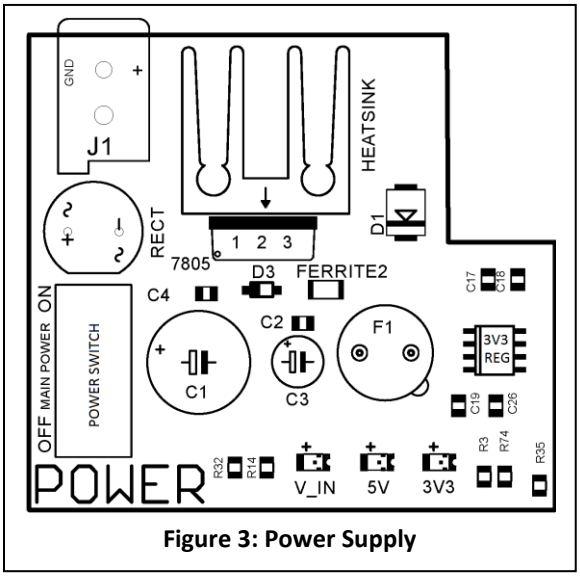

Figure 3: Power Supply

3.1 Power Supply

The Power Supply module is designed to take DC input of 7.5V to 17V, and output regulated 5V and 3.3V for all modules to share including the I/O bus. It is able to also receive 5V USB power from the host PC. Either of these sources can be used as power supply, although a dedicated DC power supply is strongly recommended. The max current supplied by the power module is limited to 1.25A in total, 3.3V and 5V rails combined. A small fuse is present to protect the power supply from operating above its current limit. (Little Fuse 372 Series, TR5 Size)

The input connector is a female 5.5×2.1mm jack, which is compatible with many commonplace adaptors. Wall adaptors of any polarity (center positive or center negative) may be used since the input is rectified.

A single two-pole slide switch controls power to the entire board by interrupting the positive power terminal immediately after rectification, as well as interrupting the positive power terminal from the USB. In the ON position, the board is powered, as indicated by both green LEDs for 5V and 3.3V; in the OFF position, all modules are unpowered. The red V_IN indicates the status at the DC input jack immidiately after rectificatoin, it will be luminated to indicate present of DC input power when the power switch is in the ON position.

3.2 On-board USB Programmer

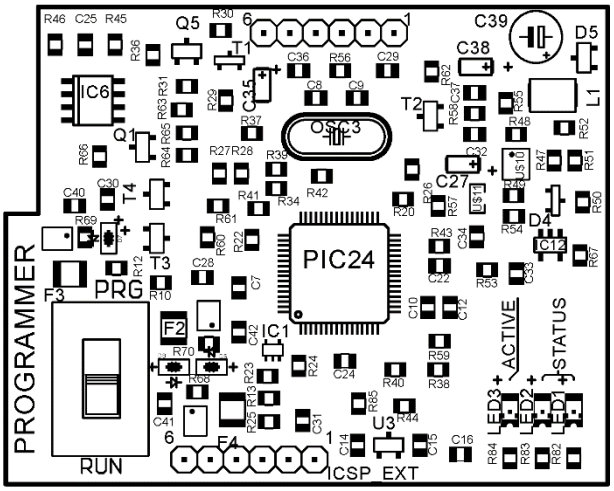

Figure 4: Onboard Programmer

The *Microcontroller board* includes an on-board USB PIC programmer. The programmer is compatible with PICkit™ 3 Debugger / Programmer and will be recognized by MPLABX IDE and IPE as a PICkit™ 3 programmer.

The programmer operates in High Voltage Programming (HVP) mode given a supply voltage of 5V. It incorporates an internal voltage converter that boosts the 5V supply to 12V needed for HVP.

To enter Programming Mode, set the slide switch to the PRG position, and to enter Executing mode, set the slide switch to the RUN position. Three LEDs indicate the status of the programmer, when the board is first power on, all LEDs should illuminate. When the programmer finishes initialization, and enters standby mode, both status LEDs will turn off leaving only the red ACTIVE LED illuminated to indicate the programmer is ready.

Note: that the onboard PICkit™ 3 does not posses the push button for the Programmer-To-Go feature, however it is possible to activate the feature if desired, see Section 5.2 for more detail.

3.3 Debugging Module

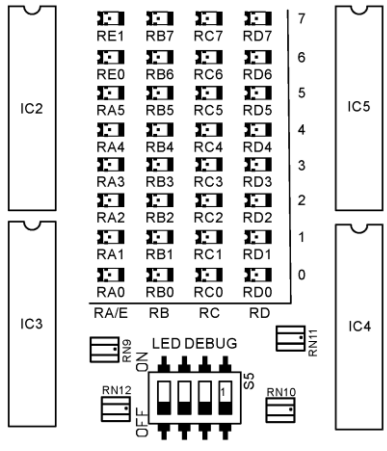

Figure 5: Indicator LEDs

The Debugging module allows the user to monitor the state of each I/O pins of the main PIC device through 32 indicator LEDs. These indicators are fully buffered, as such they do not indice any load on the signal lines in which they are monitor – in other words, they can safely monitor the logic states of low-power sensor signals. However, it should be noted that unless specific lines are connected to external signals or are driven by the I/O ports, their indicators may change unpredictably since they are floating.

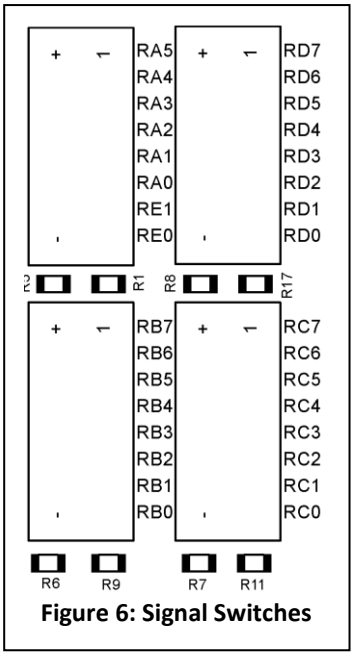

Figure 6: Signal Switches

The indicators can be disabled in four separate columns through the included (LED DEBUG) DIP switch, in order to reduce current draw on the boar's power supply when not needed. The DIP switches, from left to right, corrospond to columns: **PORTA/E PORTB PORTC PORTD** respectively.

The Microcontroller board also comes with 32 DIP switches to provide input to the pins of the main PIC for debugging purposes. Each switch corresponds to an pin on the main PIC as indicated by the marking next to each switch. Each DIP switch has 3 states: High (5V or 3V), Floating (disconnected), and Ground (0V). The switchs are in the middle position by default, which disconnects its corrosponding pin and leave it floating. This is the perferred state if the user wishes to hand control of the pin to the PIC or external circutry connected to the I/O bus. The switches can be pulled to the left or right to send a high or low state to the PIC. It is importat to be careful to ensure the pin state is not enforced by the PIC or any external input before using the signal emulation switchs. A conflict between the switchs and the PIC could lead to unexpected behaviors of the development board.

3.4 HD44780 Based LCD

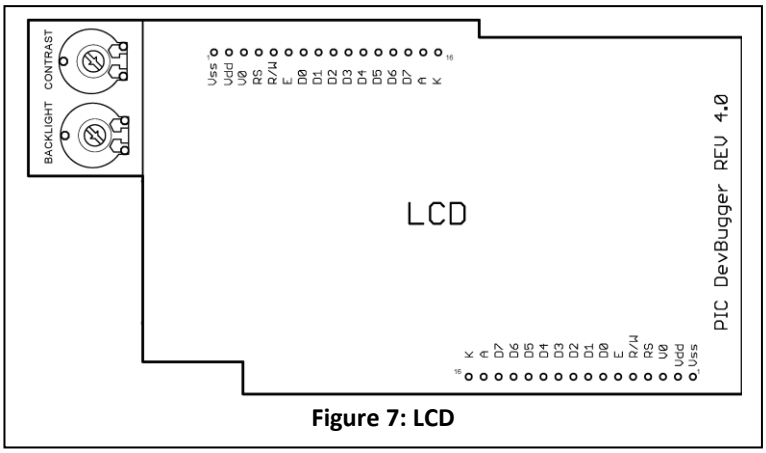

Figure 7: LCD

This peripheral module allows a HD44780-based LCD display to be easily connected to the board. Controls are provided for backlight and contrast, and two equivalent headers have been provided to support different LCD orientations (Up/Down facing).

The board have been designed to accommodate both a 2x16 character display or a larger 4x20 character display. The two numbers refer to the number or rows and number of characters per row. A 2x16 LCD will have 2 rows with 16 characters each, while a 4X20 LCD will allow 4 rows with 20 character per row. Note that 4X20 (80 characters) is the maximum possible size for the HD44780 LCD controller.

HD44780(LCD)	PIC I/O Pin
RS	RD2
R/W	GND
E	RD3
D4	RD4
D5	RD5
D6	RD6
D7	RD7

Table 1: LCD I/O Map

Since the HD44780 protocol supports either 8-bit or 4-bit data transfer modes, the *Microcontroller board* has been configured to use 4-bit mode in the interest of conserving I/O pins. The HD44780 interface pins have been mapped to the PIC I/O ports as shown in Table 1.

3.5 4x4 KEYPAD and Co-Processor

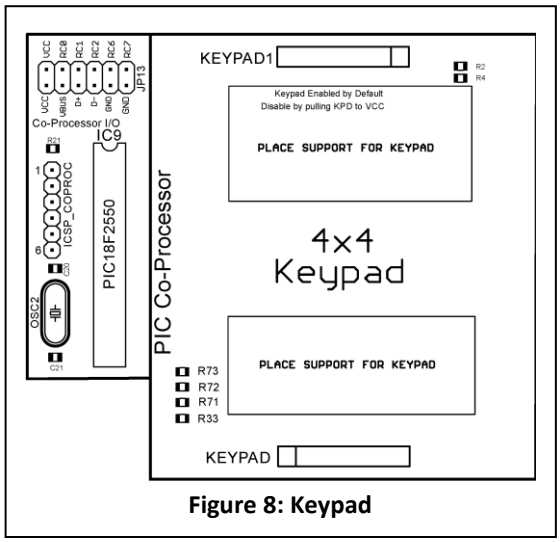

Figure 8: Keypad

4x4 matrix keypads are commonly used in microcontroller applications, this module was included to simplify the required interface. Two equivalent headers provide a socket for the keypad in different orientations, and a **PIC18F2550** microcontroller encodes the 8-line matrix input into simple to parse 4-bit hex data. The data pins for the encoder PIC are connected to **PORTB<7:4>** and the 'data available' pin (active high) is connected to **RB1**.

The Keypad module can be disabled on-the-fly through the special **KPD** pin on the I/O bus If **KPD** is set high, the keypad will be *disabled*, and if set low, the keypad will be *enabled*. When the keypad module is enabled, two 47HC4066 buffer chips isolate the keypad data pins from outputs on the PIC I/O bus. Disconnecting the keypad data pins protects them from interference from external circuitry and increase reliability of keypad inputs. When the keypad is disabled, the encoder PIC will set its output pins to high impedance mode (disconnected), also PORTB<7:4> and RB1 of the main PIC will be made accessible to the user. Note that **KPD** can be controlled either by external circuitry or directly by the PIC by connecting it to one of the ports on the I/O bus.

Additional to acting as the keypad encoder chip, the PIC18F2550 also serves as a Co-Processor to the main PIC. A dedicated programming header and oscillator socket allows it to operate independently or in conjuncture with the main PIC.

3.6 A2D Reference

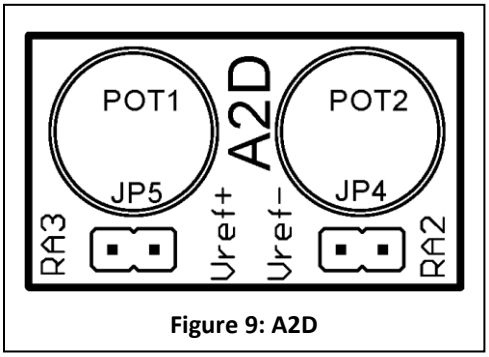

Figure 9: A2D

The *Microcontroller board* has been equipped with two potentiometers, **R12** and **R13** to set the voltage reference levels for the Analog to Digital Converter (**ADC**). To enable these references, short JP4 and JP5; if they are both left unconnected, the references are disabled.

Short **JP4** to enable Vref on **RA2**
Short **JP5** to enable Vref on **RA3**

Note: To enable voltage references, the PIC also need to be configured alongside the insertion of jumper links. Refer to data sheets of PIC device used for instructions.

3.7 Real Time Clock

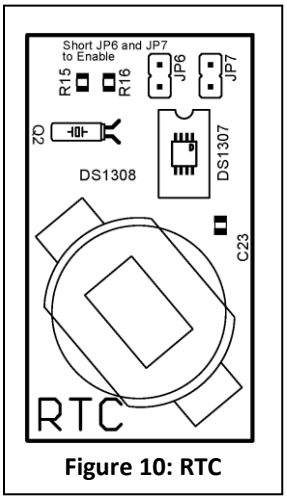

Figure 10: RTC

A Real Time Clock (**RTC**) and CR2023 coin cell battery allow for off-chip timekeeping, even when the rest of the board is unpowered. The circuit is designed for a DS1307 or DS1308 RTC chip, interfaced through I^2C on pins **RC3** and **RC4**.

To enable, short **JP6** and **JP7**

Note: A DS1307 or a DS1308 RTC chip may be included. The two chips are extremely similar in operation. The DS1308 chip come in a surface mount package, while the DS1307 chip take form of an 8 pin Dual-in-line (DIP) package, and will be supplied with a matching socket on the board. The CR2023 coin cell battery is not included with the development board.

Note 2: The RTC module was designed to be used with backup power (the 3V disk battery) and therefore must have the battery in the socket to ensure consistent operation.

3.7.1 Using the Real Time Clock

The DS1307 real time clock can be used to keep track of time in seconds, minutes, hours, days, months, and years. The numbers of days in months are automatically adjusted, including leap years. The hours function allows the chip to keep time in either 12 or 24-hour format with AM/PM indicator for 12-hour format. The advantage of the off-chip timekeeping functionality of the RTC is to free the microcontroller from the task so that it may focus on other tasks. For more detail on the DS1307 real time clock, please refer to Chapter 7 of the AER201 course notes.

The RTC communicates with the main PIC microcontroller through the I^2C protocol and acts as a slave device with a 7-bit address of 1101 000X (where X denotes if the transaction

is a read or a write in in a I2C transaction). In order to use the RTC, the main PIC must be configured as a MSSP device or master device for I^2C. The master device is responsible for initiating and controlling the clock pulse for all slave devices, including the RTC. The sample code for configuration and operation of the RTC chip is available to students as a MATLABX project: PIC18_RTC.X

The C libraries for I^2C communication is also included in the sample MPLABX project, the library can be moved to any new project and provide a basic framework for communication with I^2C peripherals.

Importing the I^2C communication libraries

To communicate with the RTC chip, the I^2C library must be first added to any new project you crate. To do this, follow the steps below:

1. Copy I2C.c and I2C.h from the RTC sample code into your project directory
2. Open MPLABX and open the project where the RTC is to be used
3. If there isn't a window displaying the list of open projects, Go to *Window* click *Projects*. (or press Control – 1)
4. In the Project window where all the files in the project are listed. Right click on the project you are working on, and select *Add Existing Item…*
5. Select and add I2C.h and I2C.c to your project.
6. In any source files in which the user code calls functions from the I^2C library, the directive *#include "I2C.h"* must be included at the start of the .c file.

Writing to RTC registers

Before beginning, the PIC must be configured as the I^2C master with an appropriate clock rate. This can be done by calling *I2C_Master_Init(10000);* which will configure the PIC as I2C master with a clock rate of 100KHz.

To write to registers of the RTC, first send a command on the I2C bus with the slave address, appended with a zero to indicate a write operation:

```
I2C_Master_Start(); //Start condition
I2C_Master_Write(0b11010000); //7 bit RTC address + Write
I2C_Master_Write(0x00); //Set location of register to write to
```

After the register address have been selected, the data intended to be written to the registers can be send one after another on the bus. The chip will automatically increment the registers after receiving each byte of data, so there is no need to restate each register address for successive writes. The code required is below:

```
for(char i=0; i<7; i++){
        I2C_Master_Write(happyNewYear[i]);
    }
I2C_Master_Stop();                    // Send stop condition
```

Where *happyNewYear* is an array that holds, in order, the register data required to set the time for the RTC. See Figure 11 for a flow chart of the data write sequence.

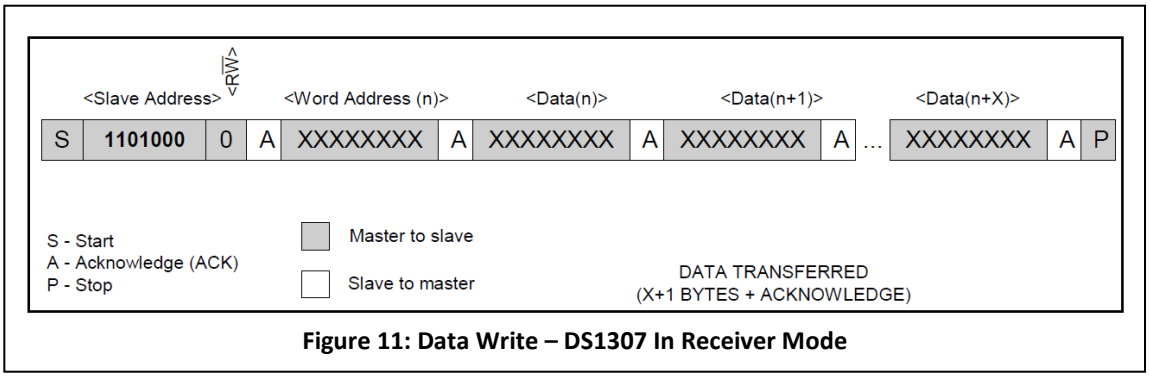

Figure 11: Data Write – DS1307 In Receiver Mode

Note: The RTC stores date and time values in BCD form in its registers, so there is no need to cast the decimal number representation into hex before sending them to the registers. For example, to set a time of: "2016 December 31 Saturday 23:59:45", simply send `{'0x45','0x59','0x23','0x07','0x31','0x12','0x16'}` to the RTC.

Reading from RTC registers

The procedure for reading values back from the RTC is very similar to the writing sequence. First, ensure the I^2C bus has been initiated using the *I2C_Master_Init(10000);* command (Initialization is only required once at the start of the program). Then send the address of the slave over the bus, appended with a '1' to indicate a read operation:

```
I2C_Master_Start();
I2C_Master_Write(0b11010001); //7 bit RTC address + Read
```

When the read bit (last bit of the address) is sent at the start of the transaction without specifying any register address, the RTC chip will send back data starting from register 0x00 continuously until the master terminated the transaction by a "not acknowledge" signal. The data can be received by continuously calling *I2C_Master_Read(1);*

```
for(unsigned char i=0;i<0x06;i++){
      time[i] = I2C_Master_Read(1); // Receive n-1 bytes with ACK
}
```

Once all except the last desired data bytes have been received, the master should receive the last byte of data following by a "not acknowledge":

```
time[6] = I2C_Master_Read(0);         //Final Read without ACK
I2C_Master_Stop();                    //Stop receiving sequence
```

Alternatively, if one does not wish to read the RTC registers starting at register 0x00, a work pointer can be first written to the RTC before initiating the receiving sequence using the following codes:

```
I2C_Master_Start(); //Start condition
I2C_Master_Write(0b11010000); //7 bit RTC address + Write
I2C_Master_Write(0x00); //Set memory pointer to seconds
I2C_Master_Stop(); //Stop condition
```

The operation flow charts for both receiving modes can be seen in Figure 12 and Figure 13.

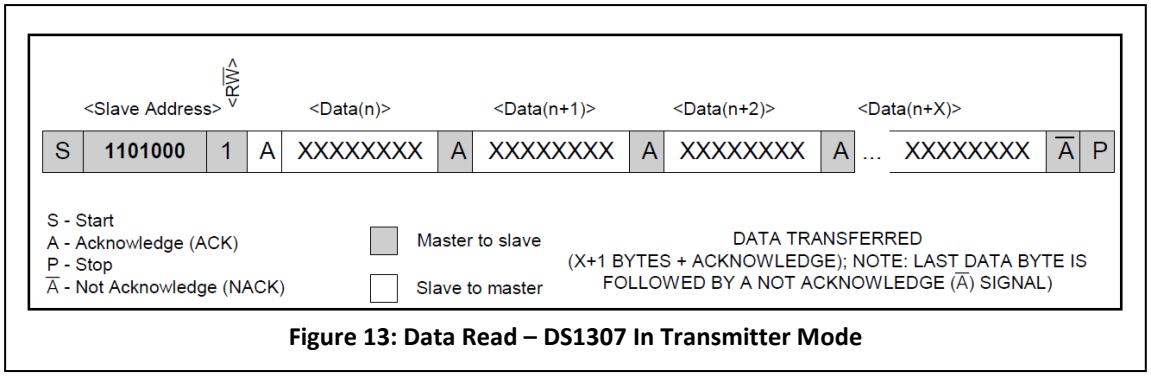

Figure 13: Data Read – DS1307 In Transmitter Mode

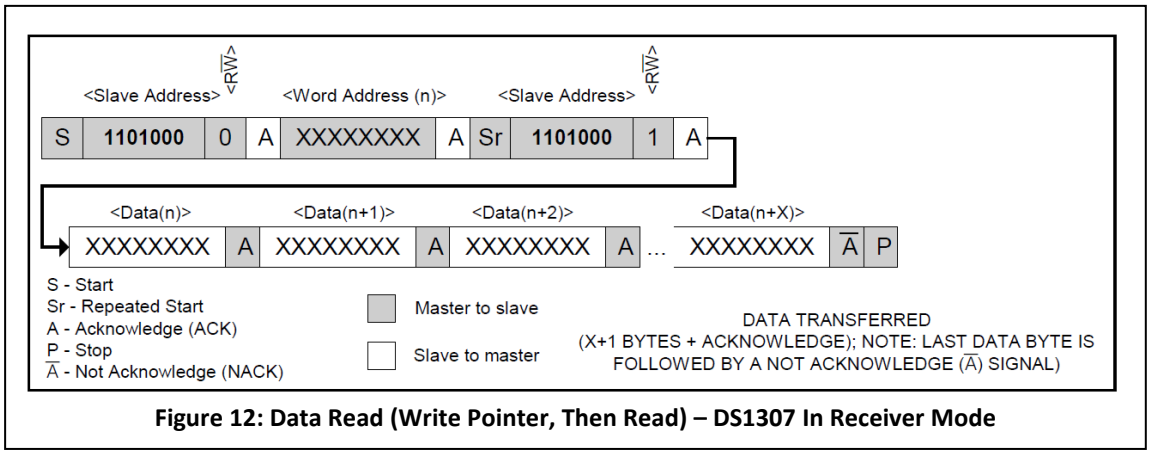

Figure 12: Data Read (Write Pointer, Then Read) – DS1307 In Receiver Mode

3.8 Main I/O Bus

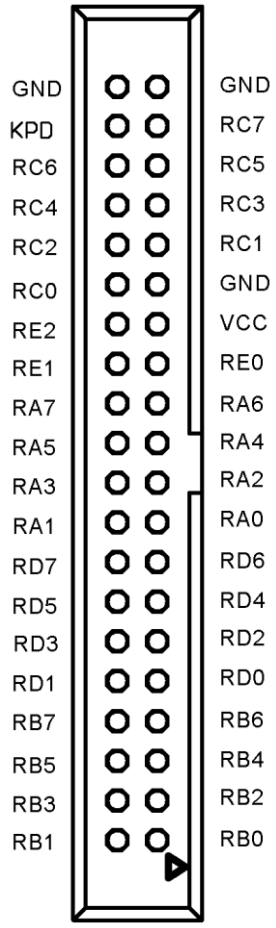

Figure 14: IO Bus Pin-Outs, Top View

A 40-pin bus has been provided to allow direct access to each I/O pin available on the PIC, as well as a special purpose pin for enabling/disabling the keypad at runtime (see Section 3.5). The pin-outs for the socket shown in Figure 14.

It is important to note that to access **RA6** and **RA7,** jumpers **JP8** and **JP9** must be properly set, as described in Section 3.10 .

Also, the user needs to note that RA6 and RA7 are not available PICs without internal oscillators, as they are dedicated oscillator pins. For PICs with internal oscillators such as PIC18s and newer generation PIC16s, RA6 and RA7 can be configured as work as general purpose IO when the internal oscillator block is used.

It is important to note that these I/O busses are connected directly to the PIC, the user should be careful to not present voltages above 5V or below 0V to any of these pins to avoid damage to the PIC chip or other peripherals.

Table 2: Pins Used by Onboard Modules below lists all pins used by peripheral modules on the development board. The user should be careful with using these pins for external I/O to avoid conflict with pin assignment with existing libraries and modules.

Module	Pins (Bolded pins can be disconnected, see section on the respective modules for more information)
RTC	**RC3, RC4**
LCD	RD2, RB3, RD4, RD5, RD6, RD7
Keypad	**RB4, RB5, RB6, RB7**, RB1
A2D	**RA2, RA3**
Programmer	**RB5, RB6, MCLR**
Arduino	**RA0, RA1, RA4, RA5, RE2, RC3, RC4, RC5, RC6, RC7, RD0:7**
Main Oscillator	**RA6, RA7**
I2C	RC3 RC4
Debug	**RE0:1, RA0:5, RB0:7, RC0:7, RD0:7**

Table 2: Pins Used by Onboard Modules

3.9 I²C BUS

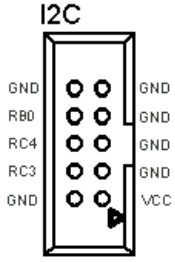

Figure 15: I2C

An I²C bus socket has also been provided to allow a separate I²C bus to a peripheral device. A 10-bin ribbon cable connector (not supplied with board) can be used for this purpose. The pinouts of the socket are shown to the left.

Note that 10K pullup resistors are included with the RTC module and is in circuit when the RTC module jumpers are inserted. It is recommended that only on set up pullup resistors be used on the I2C bus. There is no need to implement pullup resistors on external circuitry connected to this breakout header if the RTC module is in use. Otherwise, separate pullup resistors are required.

3.10 Main PIC Device

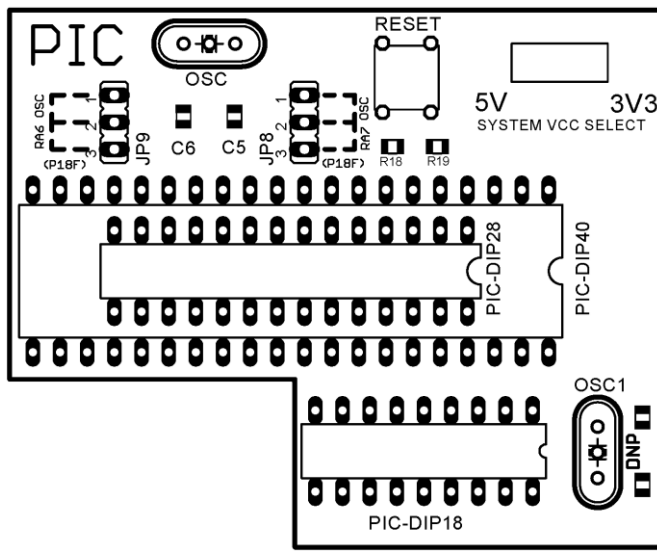

Figure 16: Main PIC Module

This section of the board has several sockets for PIC devices of different sizes. Only one socket may be occupied at a time, otherwise bus conflicts will arise. The *Microcontroller board* is primarily intended to be used with a PIC18F4620, although most other PIC devices in the **PIC16F** and **PIC18F** families are currently supported.

Additionally, some PIC16F and PIC18F devices allow the user to employ an internal oscillator and configure **RA6** and **RA7** as general purpose I/O pins. If this is intended, jumpers **JP9** and **JP8** must be set as follows:

Short pins **1 and 2** to use external oscillator
Short pins **2 and 3** to enable **RA6, RA7**

Of course, the appropriate configurations must also be set from within the code.

A switch is also present for user to select the global VCC voltage between 5V and 3.3V for low voltage compatible PICs. It is important to note that this switch is global to the board,

and if 3.3V operation is desired, the LCD, RTC and Arduino board must be switched to their 3.3V compatible alternatives.

3.11 PIC – Arduino Interface

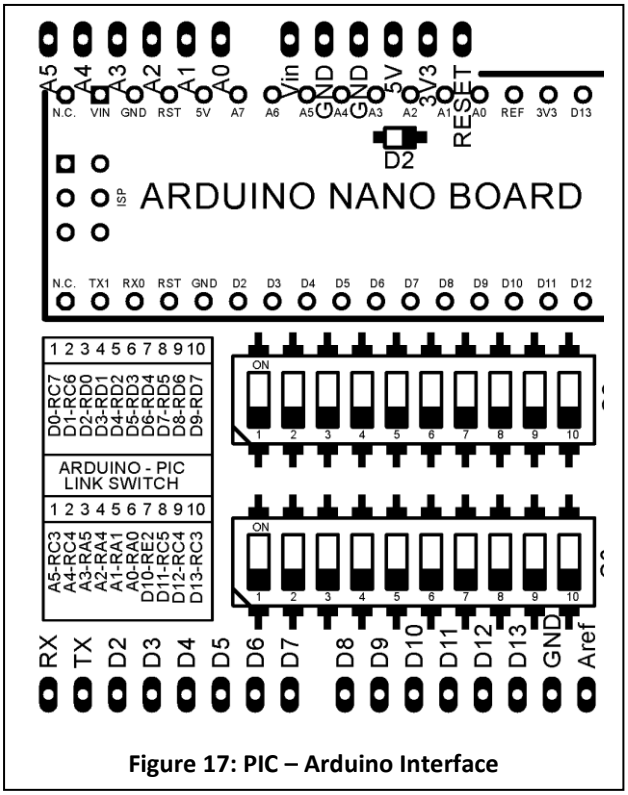

Figure 17: PIC – Arduino Interface

This section of the board allows interface with Arduino boards and shields. An Arduino Nano board can be inserted into the headers in the "ARDUINO NANO BOARD" section, and switch ban S2 and S3 enables pin to pin connection between the PIC and Arduino board and shield.

Pins on the Arduino board header is always connected to matching pins on the shield connector, regardless of the positions of the switches. As such, this module can also be used simply as a shield adapter for a standard Arduino Nano board.

5V power for the Arduino board is connected directly to the 5V power rails of the development board, and the 3.3V supply on the Arduino is connected to the 3.3V power rail through a diode that prevents current draw from the 3.3V regulators on board the Arduino board.

The PIC – Arduino connection controller by each switch are shown in the table to the left of each switch bank on the board. The pin connects are designed in such a way that the PIC and Arduino is able to communicate through various busses including Serial Peripherals Interface (SPI), Universal Asynchronous Receiver/Transmitter (UART) and I^2C. Also, the LCD control pins are connected to the GPIO pins on the Arduino. If desired, control of the LCD can be taken over by the Arduino Nano Board.

Relevant pin connections for bus communication and LCD are outlined in table # below:

Function – PIC – Arduino	Function – PIC – Arduino	Function – PIC – Arduino
LCD RS – RD2 – D4	LCD D7 – RD7 – D9	SPI SCK – RC4 – D13
LCD E – RD3 – D5	I2C SCL – RC3 – A5	SPI D0/MOSI – RC5 – D11
LCD D4 – RD4 – D6	I2C SDA – RC4 – A4	SPI DI/MISO – RC4 – D12
LCD D5 – RD5 – D7	UART A->P – RC6 – D1	SPI SS/CS – RE2 – D10
LCD D6 – RD6 – D8	UART P->A – RC7 – D0	

Note: RC3 and RC4 on the PIC are shared by both SPI and I2C modules, while the ATmega chip used by the Arduino have separate pins for these two functions.

4. Programming and Software

4.1 Overview

There is no specific programming software supplied with the microcontroller board. The onboard programmer is compatible with all MPLABX® and MPLAB® programming tools as a PICkit™ 3 Debugger/Programmer. The onboard programmer supports all PICs devices supported by the official PICkit™ 3 programmer.

Quick start and full user guides for MPLABX® IDE and MPLABX® IPE can be found from the official Microchip website. The links included below will help you get access to these files and get started promptly:

PICkit™ 3 In-Circuit Debugger/Programmer User's Guide For MPLAB® X IDE:
http://ww1.microchip.com/downloads/en/DeviceDoc/52116A.pdf

Integrated Programming Environment (IPE) User's Guide:
http://ww1.microchip.com/downloads/en/DeviceDoc/50002227A.pdf

MPLAB® X IDE User's Guide:
http://ww1.microchip.com/downloads/en/DeviceDoc/50002027D.pdf

4.2 Using the Onboard Programmer

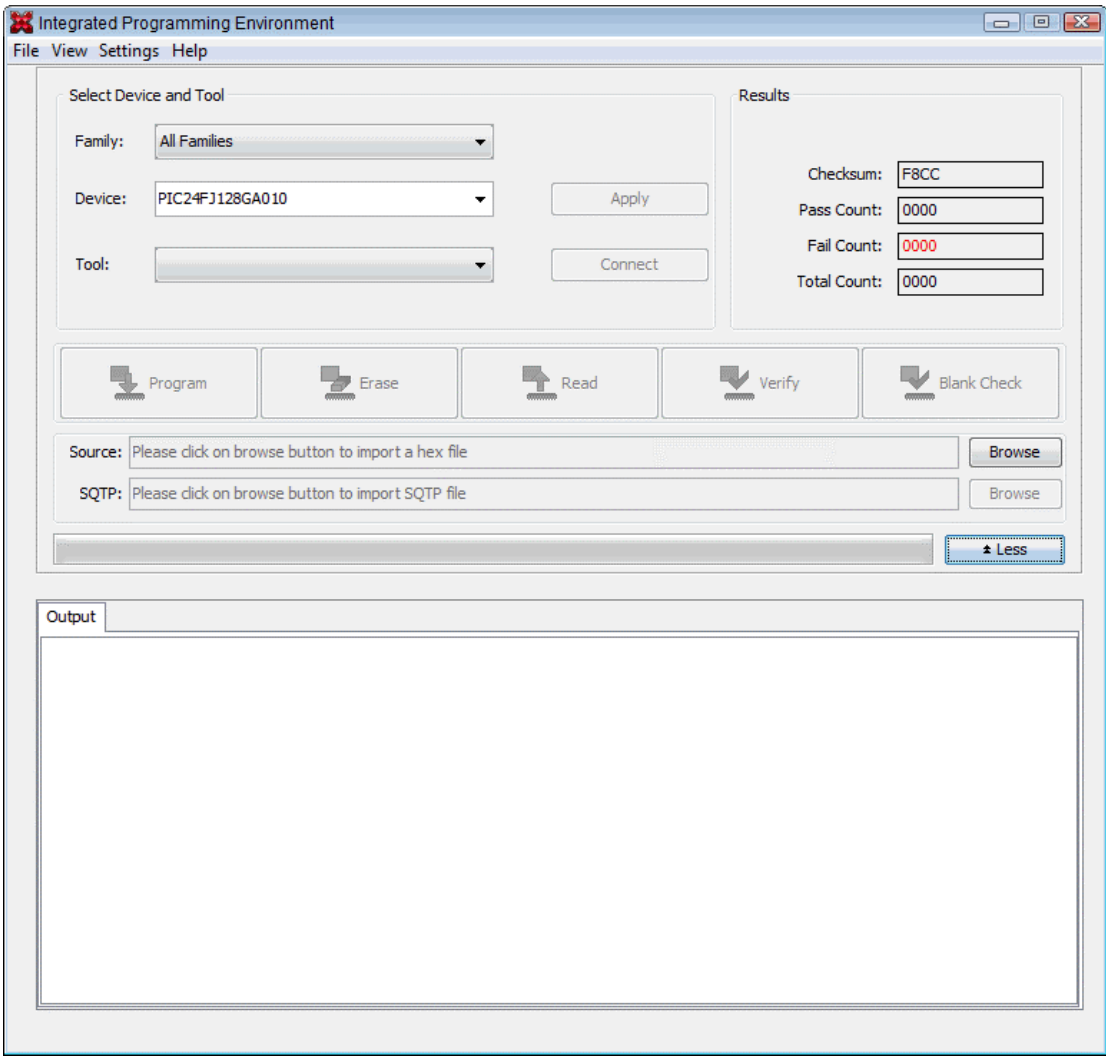

Figure 18: MPLABX® IPE Application

For those without prior experience with PIC programming software, the following steps can generally be used to load the HEX code onto the device using the MPLABX® IPE:

1. Connect the *Development Board* to the PC.
2. Flip the programming switch to PRG mode and flip the power switch to ON.
3. Select the correct Family and Device from the drop down menu. For example, If the PIC18F4620 device is used, select: Advanced 8-bit MCUs (PIC 18), PIC18F4620
4. The programmer should be detected and will be displayed as a PICkit™ 3 programmer in the tool section.
5. Click "Connect" to connect to the programmer.
6. Wait for any firmware update (if applicable) to download and install.
7. Go to File→ Import→ Hex to load a HEX file.

8. Press the **Program** button to load the HEX file to the PIC device. The IPE will indicate if programming is successful or if any error occurred.
9. The IPE will sometimes caution user to check if the device is truly a 5V device, this is normal. Since the PICkit™ 3 may cause damage to a low voltage devices if the programmer uses 5V on its I/O pins.

Note, if the Development board is disconnected at any point, it must be reconnected to the application by pressing the "Connect" button before programming.

If any errors occur or the application cannot find the programmer or the PIC device, make sure the programming switch is in PRG mode, turn off the power to reset the board, and try again.

The following is a description of the common operations that may be performed within the MPLABX IPE.

Program downloads the currently loaded HEX code into the target device.

Erase performs a bulk erase of the device, effectively returning it to its factory state. In some cases, this may fix a device that appears faulty.

Read is used to read the current program loaded into the target PIC device. This program can be retrieved and stored in a HEX file. It also retrieves the EEPROM data stored in the PIC. Note that the option to save the HEX file will not be available until it is enabled from advanced mode. For more detail on usage of the advanced mode, see MPLABX IPE User Manual, Section 2.5.

Verify reads the contents of the PIC device and compares it against the loaded HEX file. If any differences are observed, an error is given and the operation fails.

Blank Check is used to verify that the chip is indeed 'blank'; this is useful after performing an *Erase Chip* operation to verify that the operation succeeded.

4.3 Sample Codes

The development board package is accompanied with 10 sets of sample codes that demonstrate functionality of the PIC as well as onboard modules. The sample codes are written in C for the MPLAB® XC8 compiler. Microchip have been dropping support for assembly language programming, the sample code packages uses the latest supported platform in ensure compatibility for the future. For an official guide on the MPLAB® XC8 compiler, refer to the links below:

MPLAB® XC8 Getting Started Guide:
http://ww1.microchip.com/downloads/en/DeviceDoc/50002173A.pdf

MPLAB® XC8 C Compiler User's Guide:
http://ww1.microchip.com/downloads/en/DeviceDoc/50002053G.pdf

Note: The XC8 Compiler is not included as part of the MPLABX® IDE package and must be downloaded and installed separately after the MPLABX® IDE.

PIC18F4620 Samples:

All sample codes below neither uses a 10MHz external crystal oscillator or the internal oscillator block. Refer to the description text file provided with each sample project for the modules utilized and oscillator block used.

ADC — This is a project designed to demonstrate the Analog to Digital conversion features of the PIC18 device. It reads the voltage input on RA2 and RA3 and displays the 10-bit value on the LCD in hexadecimal form.

Board_Test — This is a project that testes RTC, Keypad, LCD and Debug modules of the development board. This project was used to test the development board after assembly.

I²C_Arduino — This project demonstrates the I²C and Arduino board interface features of the board. The PIC device communicates with the Arduino Nano board and send the character pressed on the keypad to a PC connected to the Arduino board. Additionally, when the A key is pressed 3 times in a row on the keypad, the LCD will turn to display characters sent to the Arduino board over Serial (9600 baud, 8-bit, no parity).

I^2C_RTC	This project demonstrates the code necessary to interface with the DS1307 real-time clock IC on the Microcontroller board, using the I^2C module of the main PIC device. The RTC time is set at the beginning and the time kept by the RTC chip is then displayed on the onboard LCD.
LCD	This project displays a message on the LCD upon startup of the PIC. The LCD library may be useful to the user.
Keypad_LCD	This project demonstrates the basics of interfacing with the Keypad and LCD modules of the Microcontroller board; anything that is typed on the keypad is immediately displayed on the LCD.
Keypad_Interrupt	This project demonstrates usage of interrupt for keypad inputs. The LCD displays the last key pressed on the keypad while the PIC executes command in an infinite loop. The keypad input is handled as an interrupt and not polled.
Port_Test	This project is a simple test program, which can be used to quickly verify MPLABX and the programmer module are functioning correctly. If this program executes correctly, then each of the Debug LED's should flash sequentially when the board is placed in RUN mode. Note that RA4 on PIC16 devices will not turn on unless pulled up by either a debug switch or an external resistor, since it is an open-drain output.
PWM	This project tests the PWM module of the PIC18F4620. It generates a 50kHz PWM signal with varying duty cycle. The signal is output through the CCP1 pin (RC2). When running, the RC2 debug LED should gradually increase in brightness before turning dark.
Keypad_Firmware	This project demonstrates reading and encoding of a 4x4 matrix keypad. This code is used on the keypad encoder PIC to encode the raw 8 pin keypad input into 5 pins for the main PIC. This code also demonstrates internal oscillator usage of the PIC18F2550.

5. More Advanced Operations

5.1 Using the Board as a Standalone Programmer

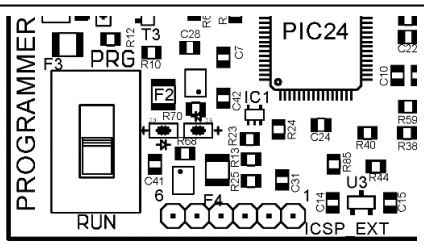

Figure 19: ICSP_EXT header

The onboard programmer on the development board have been designed with the capability to program external PIC devices though the ICSP interface. The ICSP interface is common for all PIC devices all the way from PIC10 to PIC32. The Pin-out of this header from right to left is shown in Table 3:

Pin #	Function
1	V_{PP}/MCLR
2	V_{DD} (VCC)
3	V_{SS} (GND)
4	ICSPDAT/PGD
5	ICSPCLK/PGC
6	ICSOLVP/PGM

Table 3 ICSP Pins

There is no need to remove the PIC device installed on the development board, however the Program/Run switch must be switch to the RUN position to isolate the onboard PIC from the programmer before the programmer can be used as a standalone programmer.

To use the programmer externally, first turn off the board, and connect the ICSP interface to the target device when the board is turned off. Then, turn on power to the development board, ensuring that the PROG/RUN switch remains the in the RUN position throughout the operation. Plug in the USB cable and use the programmer as you would the same way for programming a PIC normally, with the exception to never move the switch to PRG position.

Note: The onboard programmer has input protection on each of the programming data pins, the ICSP pins will become inoperable for some time if a large amount of current is sourced or sunk from it. If the programmer is failing to detect the device despite proper setting of the program/run switch, turn off the board and let it rest for a minute or so before attempting to program again.

5.2 Activating the Programmer-To-Go Feature

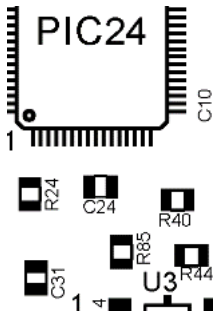

Figure 20: Position of R85

The onboard programmer does not have a pushbutton to activate the Programmer-To-Go feature that is part of the PICkit™ 3 programmer. However, all other hardware required for Programmer-To-Go are all present as part of the programmer, so the feature can still be accessed if the user requires it. To simulate the push button operation, take a fine tipped conductor, such as a jumper cable, connect one end of the cable to GND and touch the other end of the cable to the open solder pad on the top side of R85 (situated just under and to the right of the PIC24 chip). This will be recognized by the PICkit™ 3 as a button push. This action is not recommended by the end user as the board is not designed for this operation and it is easy to make mistakes and short other pins to ground. The procedure for loading code into the PICkit™ 3 for Programmer-To-Go is not covered here.

5.3 Re-imaging Firmware for the Onboard Programmer

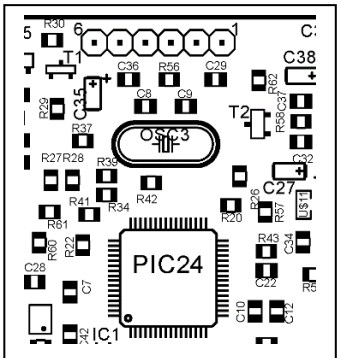

Figure 21: Position of ICSP_PK3

The development board is fully programmed and ready to use out of the box. However, it has been observed in previous iterations of this development board that many users experienced firmware faults on the programmer that required the programmer firmware to be re-imaged.

To reimage the programmer PIC, plug in an external PICkit 3 programmer (PICkit 2 support for PIC24FJ parts are not very reliable) to the ICSP_PK3 header on the very top of the programmer module. The pins of the ICSP header are arranged such that pin 6 is on the very left edge, and pin 1 is on the right most corner. When a PICkit 3 is used, the white tringle making on the PICkit 3 body indicates pin 1 of the ICSP interface. When inserted, the indicator LEDs should face toward the rest of the board.

The firmware file is provided as part of the sample code package along with the development board, inside the "Production Firmware" folder. It is named "DB4_PK3_24FJ256GB106_Firmware.hex".

To reimage the programmer, follow the below procedures:

1. Open MPLABX Independent Programming Environment (IPE)
2. Connect PICKit3 Programmer to the computer
3. Select 16-bit MCUs (PIC24) in the Family dropdown menu
4. Select PIC24FJ256GB106 in the device dropdown menu

5. The IPE should have detected the PICKit3 programmer, click connect to connect to the programmer.
6. Wait for the IPE to download and update firmware for the PICKit3. Reconnect to the programmer if the procedure interrupts itself. (This will take some time)
7. Click File -> Import -> Hex. Navigate to and select the file "DB4_PK3_24FJ256GB106_Firmware.hex" to load it.
8. Ensure the board being reprogrammed is powered on, and the program/run switch is in the "RUN" position.
9. Click "Program" on the main interface, and wait for programming to finish.
10. The programmer firmware should now be restored to their default firmware.

To verify the programmer firmware have been correctly loaded, detach the programmer and cycle power on the board just programmed. The 3 indicators LEDs should all light up when the board is first powered, and after approximately 5 seconds, the 2 status LED will extinguish, leaving only the red active LED illuminated. If this behavior is observed, the programmer is properly loaded and functioning correctly.

5.4 Re-imaging Firmware for the Keypad Encoder PIC

Reimaging procedure for the Keypad Encoder PIC is very similar to the reimaging procedure for the programmer. The ICSP header for the Co-Processor can be found just to the left of the PIC18F2550 chip on the keypad module. The firmware for the PIC encoder chip can be found as one of the sample projects, or alternatively as "DB4_KeypadEnc_18F2550_Firmware.hex" inside the "Production Firmware" folder.

APPENDIX 1: List of all Removable Parts

The following table contains all the removable parts from the development board and their respective part numbers.

Part	Description	Part number (Digi-Key)
DS1307 RTC Chip	DS1307 RTC chip	DS1307+-ND
Jumper Links RTC x2	Short jumpers, for JP6 and JP7, without grip	S9001-ND
Rectifier 2W04G	4pin, cylindrical	2W04G-E4/51GI-ND
Fuse Cylindrical, Series 373	Red cylindrical fuse. Ensure Series number is 373	WK3050BK-ND
Crystal 10MHZ	10MHZ crystal, primary oscillator, 20pF	CTX1058-ND
Crystal Jumpers x2	Jumper links for JP8 and JP9, with grip	S9341-ND
PIC18F4620, 40Pin package	Large (40Pin) PIC18F420 chip	PIC18F4620-I/P-ND
Keypad Buffer 74HC4066 x2	14pin, 74HC4066 chips	296-14533-5-ND
Keypad Encoder PIC18F2550	Small (28Pin) PIC18F2550 chip	PIC18F2550-I/SP-ND
Debug Buffers x4	20Pin, 74HC244 Buffer Chip	296-1582-5-ND
Keypad	4x4 keypad	GH5004-ND
LCD	LCD display 16x2 or 40x2	NHD-0216BZ-RN-YBW-ND
RTC Battery CR2032	Coin cell battery for the RTC, CR2032	SY189-ND

Note: The part numbers included are for reference only, and the parts installed on the board might not be the exact part listed by Digi key.