Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

1300 Henley Court
Pullman, WA 99163
509.334.6306
www.store.digilent.com

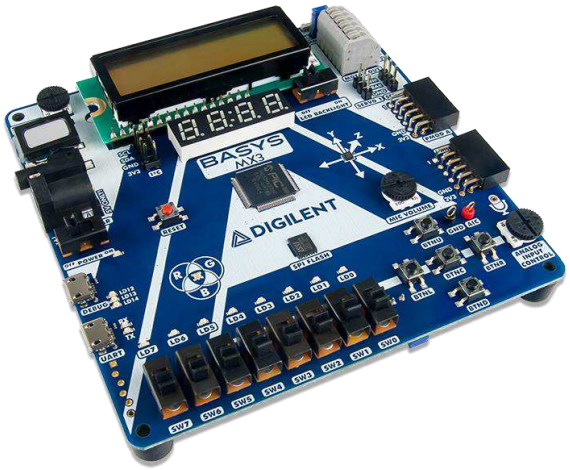
Basys MX3™ Board Reference Manual

Revised April 21, 2017
This manual applies to the Basys MX3 rev. B

Table of Contents

Overview

The Basys MX3 is a true MCU trainer board designed from the ground up around the teaching experience. Basys MX3 features the PIC32MX370 from Microchip and was designed to be used with the MPLAB® X IDE. With an exhaustive set of peripherals, students gain exposure to a wide range of embedded systems related concepts while using a professional grade tool set. Accompanied by free and open source coursework, including seven in-depth teaching units and 15 complete labs, the Basys MX3 is a versatile MCU trainer board ideal for teaching introductory embedded systems courses.



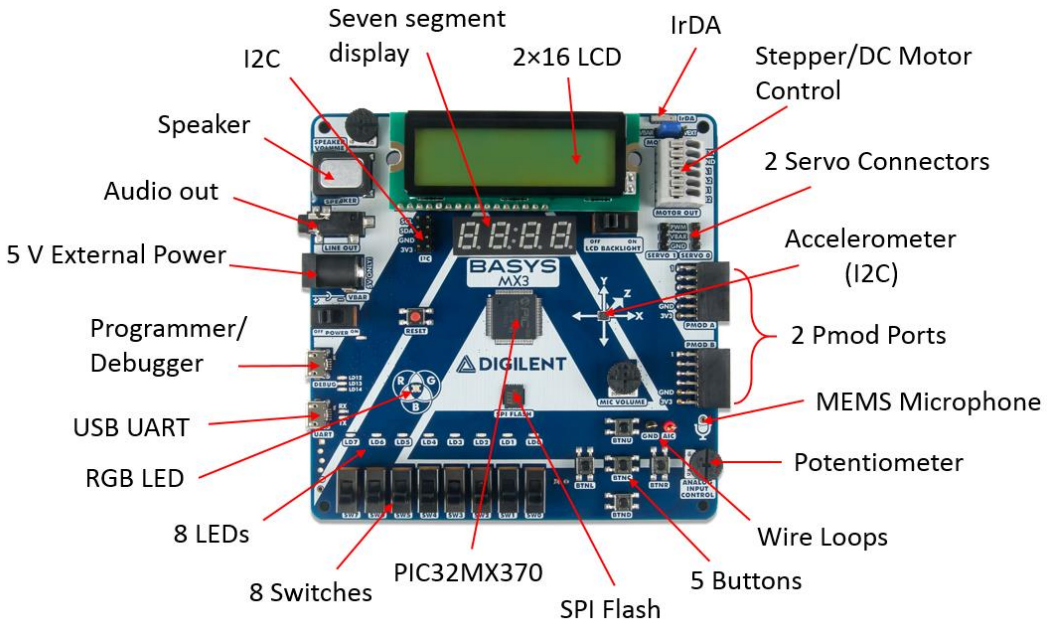
The Basys MX3.

- **PIC32MX370F512L Microcontroller**
  - MIPS32® M4K® core runs up to 96 MHz using onboard 8 MHz oscillator
  - 512 KB of Program Flash Memory, 12 KB of Boot Flash Memory
  - 128 KB of SRAM
  - Four Direct Memory Access (DMA) Modules
  - Two SPI, Two I²C, and Five UART serial interfaces
  - Parallel Master Port (PMP) for graphics interfaces
  - Five 16-bit Timers/Counters
  - Five Input Capture Modules
  - Five Output Compare Modules
  - 85 I/O pins
    - 54 pins support Peripheral Pin Select (PPS) for function remapping

- Power:
  - Powered from USB or any 5V external power source
- USB and Debugging:
  - USB-UART Bridge
  - USB programmer/debugger
  - 30-pin Analog Discovery 2 connector

- Switches, Push-buttons, LEDs, and Displays:
  - 5 Push-buttons
  - 1 Reset button
  - 8 Slide switches
  - 8 LEDs
  - 1 RGB LED
  - 4-digit 7-segment display
  - LCD character display
- Expansion Connectors:
  - Two standard Pmod ports
    - 16 Total microcontroller I/O
  - One I2C Connector
    - 2 Total microcontroller I/O

- Audio, Motor Control, and Other Devices:
  - Speaker with Audio Output Jack and volume control
  - Microphone with volume control
  - Dual H-Bridge Motor Driver for up to two 1.5 A Brushed DC Motors or one stepper motor
  - 2 Servo Connectors
  - FIR-compatible IrDA Module
  - Potentiometer
  - 3-axis, 12-bit accelerometer
  - 4 MB SPI Flash



Software Support

The Basys MX3 is fully supported by Microchip's MPLAB X IDE. See section 1 on Programming the Board for more information on using the Basys MX3 in MPLAB X IDE. Digilent provides a set of libraries called the Basys MX3 Library Pack that adds support for all onboard peripherals. This library pack can be downloaded from the Basys MX3 Resource Center.

The Basys MX3 can also be used in Arduino IDE once the Digilent Core for Arduino IDE has been installed. Instructions for installing the Digilent Core for Arduino IDE can be found on the Basys MX3 Resource Center.

Coursework and Additional Materials

Basys MX3 comes with a complete set of coursework designed to give teaching professionals flexibility in designing embedded systems and other microprocessor courses. With almost 300 pages of material, "Embedded Systems Basys MX3 and PIC32MX370" covers topics from toggling LEDs, motor control, and introduction to digital signal processing. Access to the full coursework is available on the Basys MX3 Resource Center.

Links to additional materials from Digilent and Microchip, including the Basys MX3 schematic and the PIC32MX370F512L datasheet, can also be found on the Basys MX3 Resource Center.

The Basys MX3 uses a lot of devices to implement all of the functionality it provides (accelerometer, flash memory, motor driver, IRDA, etc.). The manufacturers of each of these devices provide detailed descriptions of their functionality in their datasheets.

1 Programming the Board

1.1 Programming Tools

The Basys MX3 can be used with Microchip's standard MPLAB X IDE. This software suite can be downloaded for free from the Microchip website and includes a free evaluation copy of the XC32 compiler for use with the PIC32 microcontroller family.

MPLAB X IDE is the tool used to write, compile, program, and debug code running on the Basys MX3 board. Programming and debugging a program on the Basys MX3 using the MPLAB X IDE is possible using the DEBUG USB connector. The board contains all the required circuitry for MPLAB X to communicate with the onboard PIC32, so no additional programming tools need to be purchased.

When creating a new project in MPLAB X, a wizard allows you to setup the environment and device specific tools. The steps for this include the following:

1. Select Microchip Embedded / Standalone Project, then use the "Select Device" option to specify the PIC32 microcontroller being used: PIC32MX370F512L.
2. Select the programming tool named Basys MX3 corresponding to the board you want to program, under Licensed Debugger group.

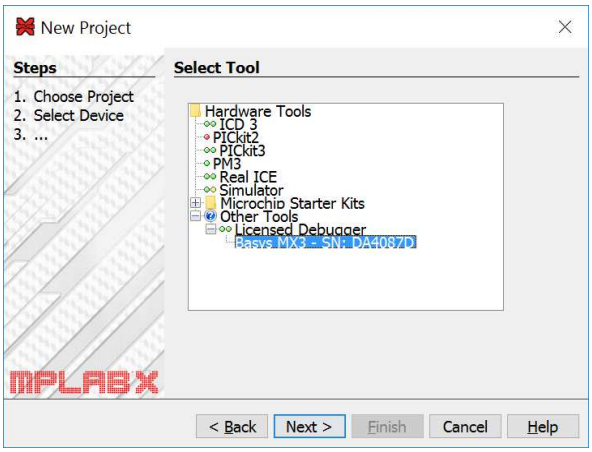


Figure 1.1. MPLAB X tool selection.

3. Select the compiler you want to use.

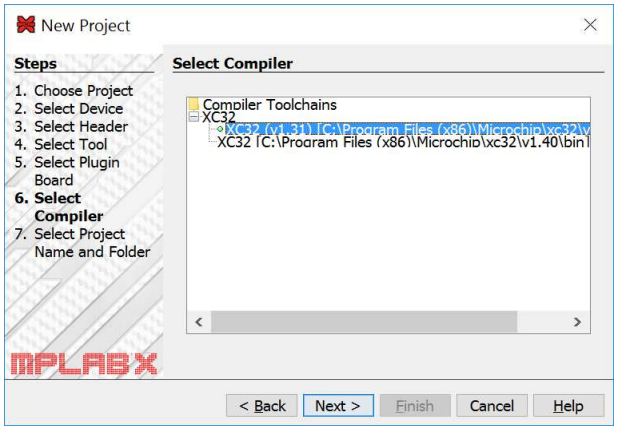


Figure 1.2. MPLAB X compiler selection.

Another useful tool included with MPLAB X is MPLAB X IPE. This tool allows the direct programming/erasing of the microcontroller, but it does not provide an environment for writing, compiling, or debugging the code. Please see Microchip documentation for instructions on using this tool.

1.2 Programming Basics

It is often very helpful to include the xc.h header when writing code for the Basys MX3:

```
#include <xc.h>
```

This further provides the inclusion of another header (p32mx370f512l.h) into the project that provides useful definitions such as:

- Register names
    - example (register LATA is set to 0):

      ```
      LATA = 0;
      ```

- Specific register bits that can be accessed using a structure having the name of the register suffixed by "bits".
    - example (bit LATA1 of the register LATA is set to 1):

      ```
      LATAbits.LATA1 = 1;
      ```

Digilent provides a set of libraries called the Basys MX3 Library Pack that addresses much of the functionality on the Basys MX3:

- ACL (accelerometer)
- ADC (analog-to-digital converter)
- AUDIO
- BTN (buttons)
- IRDA
- LCD
- LED
- MIC (microphone)
- MOT (motors)
- PMODS
- RGBLED
- SPIFLASH
- SSD
- SWT (switches)
- UART

These libraries are wrappers over the lower level functions that access the registers, allowing the user to call the functionality using functions like:

```
LED_Init();

LED_SetValue(4, 1); //turn on LED4
```

This set of libraries comes with the user documentation, but this is what you must do in order to use them:

- Include in your project the .c and .h files corresponding to the module you want to use (for example led.c and led.h).
- In your code, include the header of the module:

```
#include "led.h"
```

- In your code, call the needed functions

1.3 Digital Inputs and Outputs

The PIC32MX370F512L microcontroller offers access to all the board resources through its pins, so understanding how to access their features is very important. The list that describes each pin functionality is included in Appendix 3. You can see that each pin may have multiple functions, but all pins have one feature in common: they have an associated digital I/O (input/output) bit. On PIC32 microcontrollers, the I/O pins are grouped into I/O Ports and are accessed via peripheral registers in the microcontroller. There are seven I/O Ports numbered A–G and each is 16 bits wide. Depending on the PIC32 microcontroller, some of the I/O Ports are not present, and not all 16 bits are accessible in all I/O Ports.

Each I/O Port has the following control registers: TRIS, LAT, PORT, ANSEL, CNPU, CNPD, and ODC. The registers for each I/O Port are named after it: TRISx, LATx, PORTx, ANSELx, CNPUx, CNPDx and ODCx. For example, port A will have the following assigned registers: TRISA, LATA, etc.

The TRIS register is used to set the pin direction. Setting a TRIS bit to 0 makes the corresponding pin an output. Setting the TRIS bit to 1 makes the pin an input.

The LAT register is used to write to the I/O Port. Writing to the LAT register sets any pins configured as outputs. Reading from the LAT register returns the last value written.

The PORT register is used to read from the I/O Port. Reading from the PORT register returns the current state of all the pins in the I/O Port. Writing to the PORT register may not produce the expected result, therefore writing to LAT register is recommended.

To summarize: write using LAT, read using PORT.

PIC32 microcontrollers allow any pin set as an output to be configured as either a normal digital output or as an open-drain output. The ODC register is used to control the output type. Setting an ODC bit to 0 makes the pin a normal output and setting it to 1 makes the pin an open-drain output.

The multifunction pins that include analog input functionality need to be configured in order to be used as digital pins by clearing the corresponding bit from ANSEL register. These pins will include ANx in their name. For example: AN11/PMA12/RB11 for RB11.

This microcontroller has a weak pull-up and a weak pull-down connected to each pin. These pull-ups and pull-downs are enabled/disabled by setting the corresponding bits from CNPU and CNPD registers to 1/0. The default setting is 0 (pull-ups and pull-downs disabled).

You can see a typical example of I/O pin configuration as output and digital output operations in the User LEDs section.

You can see a typical example of I/O pin configuration as input (including analog disable) and digital input operations in the User Buttons section.

Refer to the PIC32MX3XX/4XX Family Datasheet, and the PIC32 Family Reference Manual, Section 12, IO Ports, for more detailed information about the operation of the I/O Ports in the microcontroller.

1.4 Remappable Pins

Users may independently map the input and/or output of most digital peripherals to a fixed subset of digital I/O pins. Pins that support the peripheral pin select feature include the designation “RPn” in their full pin designation, where “RP” designates a remappable peripheral and “n” is the remappable port number.

The available peripherals to be mapped are digital-only. These include general serial communications (UART and SPI), general purpose timer clock inputs, timer-related peripherals (input capture and output compare), and interrupt-on-change inputs.

On the other hand, some peripheral modules cannot be included in the peripheral pin select feature because it requires special I/O circuitry on a specific port and it cannot be easily connected to multiple pins. These modules include I2C and analog-to-digital converters (ADC), among others.

Peripheral pin select features are controlled using two sets of Special Function Registers (SFRs): one to map peripheral inputs, and one to map peripheral outputs.

The peripheral inputs are mapped and named from the peripheral perspective (based on the peripheral). The [pin name]R registers, where [pin name] refers to the specific peripheral pins, are used to configure peripheral input mapping. TABLE 12-1 in the PIC32MX370F512L datasheet from Microchip (and Appendix 1 in this document) shows the different pins and their values available to assign to a peripheral pin.

The following example shows how different I/Os, such as pin RF4, can be assigned to U1RX input pin of the UART1 peripheral:

```
U1RXR = 0x02; // 0010 corresponds to RF4
```

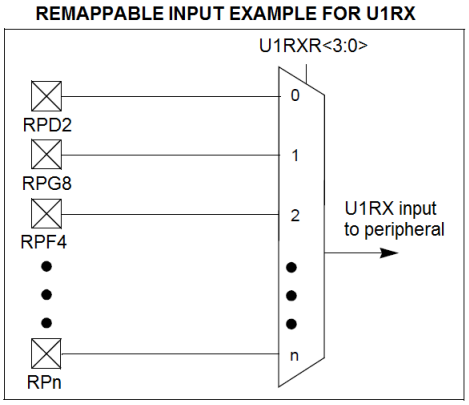


Figure 1.3. Remappable input example.

The peripheral outputs are mapped and named from the pin perspective (on the basis of the pin). The RPnR registers (Register 12-2) are used to control output mapping. The PIC32MX370F512L datasheet details in TABLE 12-2 (and Appendix 2 in this document) the values corresponding to each IO pin, associated to each available peripheral pin. Note that the current version of the Basys MX3 schematic (B.0) incorrectly lists RD6 and RD7 as remappable pins (RPD6 and RPD7, respectively); these pins are not remappable on the PIC32MX370F512L.

The following example shows how different peripheral outputs, such as U3TX, can be assigned to pin RF4:

```
RPF4R = 0x01; // 0001 corresponds to U3TX
```

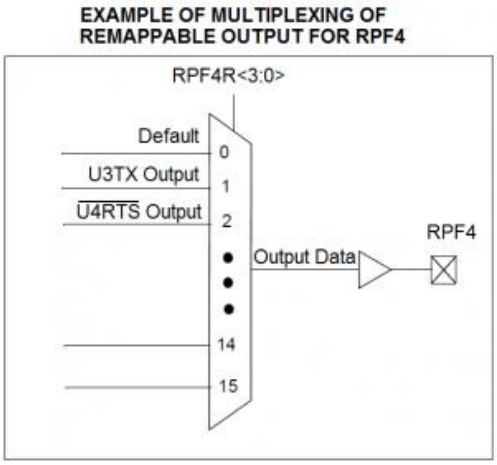


Figure 1.4. Remappable output example.

Input and output remapping is illustrated in the SPI2 section, where the SPI2 pins are mapped over the pins of PMOD A connector.

1.5 CPU Clock Source

The PIC32 microcontroller supports numerous clock source options for the main processor operating clock. The Basys MX3 uses an 8 MHz external crystal for use with the XT oscillator option. Oscillator options are selected via the configuration settings specified using the `#pragma config` statement. Use `#pragma config POSCMOD=XT` to select the XT option.

Using the internal system clock phase-locked loop (PLL), it is possible to select numerous multiples or divisions of the 8 MHz oscillator to produce CPU operating frequencies up to 80 MHz. The clock circuit PLL provides an input divider, multiplier, and output divider. The external clock frequency (8 MHz) is first divided by the input divider value selected. This is multiplied by the selected multiplier value and then finally divided by the selected output divider. The result is the system clock, SYSCLK, frequency. The SYSCLK frequency is used by the CPU, DMA controller, interrupt controller, and pre-fetch cache.

The values controlling the operating frequency are specified using the PIC32MX370 configuration variables. These are set using the `#pragma config` statement. Use `#pragma config FPLLIDIV` to set the input divider, `#pragma config FPLLMUL` to set the multiplication factor, and `#pragma config FPLLODIV` to set the output divider. Refer to the PIC32MX3XX/4XX Family Datasheet and the PIC32MX Family Reference Manual, Section Oscillators, for information on how to choose the correct values, as not all combinations of multiplication and division factors will work.

In addition to configuring the SYSCLK frequency, the peripheral bus clock, PBCLK, frequency is also configurable. The peripheral bus clock is used for most peripheral devices; particularly the clock used by the timers and serial controllers (UART, SPI, I2C). The PBLCK frequency is a division of the SYSCLK frequency selected using `#pragma config FPBDIV`. The PBCLK divider can be set to divide by 1, 2, 4, or 8.

The following example will set up the Basys MX3 for operation with a SYSCLK frequency of 80 MHz and a PBCLK frequency of 80 MHz.

```
#pragma config FNOSC =  FRCPLL
#pragma config FSOSCEN =      OFF
```

```
#pragma config POSCMOD =      XT
#pragma config OSCIOFNC =     ON
#pragma config FPBDIV =     DIV_1

#pragma config FPLLIDIV =     DIV_2
#pragma config FPLLMUL =      MUL_20
#pragma config FPLLODIV =     DIV_1
```

2 Power Supplies

The Basys MX3 requires a 5V power source to operate. This power source can come from the Programming / Debugging USB port (J12), the USB-UART (J10), or from an external 5V DC power supply that's connected to Power Jack (J11). These three power inputs are connected through Schottky diodes to form the primary input power network, VIN, which is used to power the onboard regulators and the majority of the onboard peripherals. No jumper is required to select the input power source. The board will automatically power on while the Power Switch (SW8) is in the on position and power is present on any of the power inputs.

A power-good LED (LD11), driven by the output of the 3.3V regulator (LMR10515), indicates that the board is receiving power and that the onboard supplies are functioning as expected. An overview of the Basys MX3 power circuit is shown in Fig 2.1.

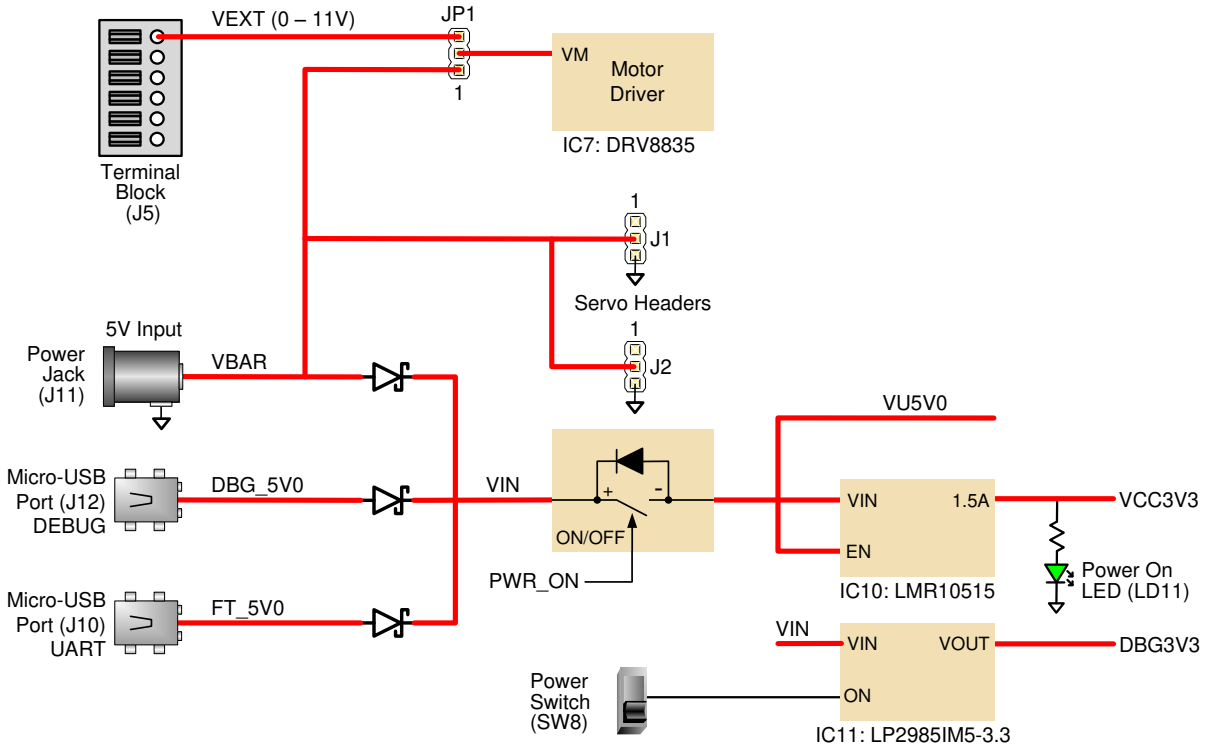


Figure 2.1. Power supply circuit.

The USB port(s) can deliver enough power for most designs; however, a few demanding applications, including any that drive multiple peripheral boards, may require more power than a USB port can provide. In these instances, an external power supply can be used. Due to their high current demands, motors and servos cannot be powered through either of the USB ports, and may only be powered through an external supply.

An external power supply can be used by plugging into Power Jack J11. The supply must use a coaxial, center-positive 2.0 mm internal-diameter plug, and provide a voltage of 5V DC (4.75V minimum, 5.5V maximum). The supply should provide a minimum current of 2A if servos are to be used. Ideally, the supply should be capable of provide 20 Watts of power (5V DC, 4A). Many suitable supplies can be purchased from Digilent or other catalog vendors.

The onboard motor driver (Texas Instruments DRV8835) may be powered by a 5V supply connected to Power Jack J11, or by an external supply (0V-11V) connected to pins 1 and 2 of Terminal Block J5. Jumper JP1 is used to select which power source is used by the motor driver.

Supply	Circuits	Device	Current (max/typical)
3.3V (VCC3V3)	PIC32MX370, PMODs, and all onboard peripherals excluding the LCD backlight, RGB LED, IrDA LED, Servos, and Motors	IC10: Texas Instruments LMR10515	1.5A/NA
3.3V (DBG3V3)	Onboard Microchip Programmer/Debugger	IC11: Texas Instruments LP2985IM5-3.3	150mA/NA

Table 2.1. Power rail characteristics.

3 User LEDs

Eight LEDs are provided, labeled LD0 – LD7 on the board (and LED0 – LED7 on the schematic), attached to eight digital I/O pins. Controlling the LEDs is done by basic access to an output I/O pin. Read more details in the Digital Inputs and Outputs section.

Figure 3.1 shows the way the LEDs are electrically connected on the Basys MX3.

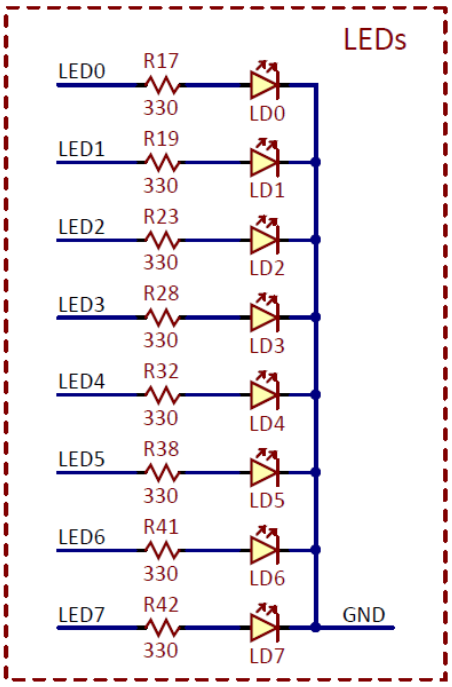


Figure 3.1. LED schematic diagram.

3.1 Connectivity

Label	Schematic name	PIC32 pin	Description
LD0	LED0	TMS/CTED1/RA0	Led 0
LD1	LED1	TCK/CTED2/RA1	Led 1
LD2	LED2	SCL2/RA2	Led 2
LD3	LED3	SDA2/RA3	Led 3
LD4	LED4	TDI/CTED9/RA4	Led 4
LD5	LED5	TDO/RA5	Led 5
LD6	LED6	TRCLK/RA6	Led 6
LD7	LED7	TRD3/CTED8/RA7	Led 7

Table 3.1. LED connectivity.

All the pins must be defined as digital output (their corresponding TRIS bit must be set to 0):

```
TRISAbits.TRISA<0-7> = 0; // LED<0-7> configured as output
```

3.2 Functionality

To turn an LED on or off, turn the corresponding digital output pin high or low by writing 1 or 0 to the corresponding LATA register bit.

```
LATAbits.LATA<0-7> = 1; // turn led on
```

or

```
LATAbits.LATA<0-7> = 0; // turn led off
```

Library functions for using the LEDs are contained in the Basys MX3 library pack, LED library; however, the user can easily use the LEDs without the LED library, as presented above.

4 User Switches

Eight switches are provided, labeled SW0 – SW7 on the board and in the schematic, attached to eight digital I/O pins of the PIC32. Reading the switches is done by basic access to an input I/O pin. Read more details in Digital Inputs and Outputs section.

Figure 4.1 shows the way the switches are electrically connected on the Basys MX3.

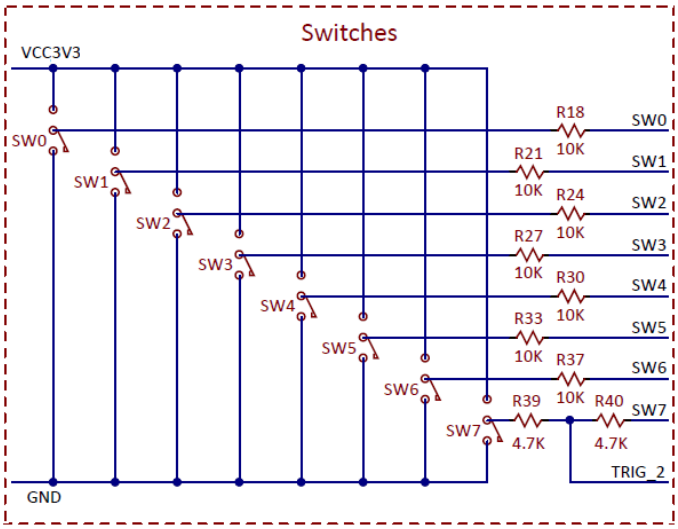


Figure 4.1. Switches schematic diagram.

4.1 Connectivity

Label	Schematic Name	PIC32 Pin	Pin Shared With	Description
SW0	SW0	RPF3/RF3		Switch 0
SW1	SW1	RPF5/PMA8/RF5		Switch 1
SW2	SW2	RPF4/PMA9/RF4		Switch 2
SW3	SW3	RPD15/RD15		Switch 3
SW4	SW4	RPD14/RD14		Switch 4
SW5	SW5	AN11/PMA12/RB11		Switch 5
SW6	SW6	CVREFOUT/AN10/RPB10/CTED11PMA13/RB10		Switch 6
SW7	SW7	AN9/RPB9/CTED4/RB9	TRIG_2	Switch 7

Table 4.1. Switches connectivity.

All the pins must be defined as digital input: their corresponding TRIS bit must be set to 1, and analog function must be disabled for pins routed to SW5, SW6, and SW7.

```
TRISFbits.TRISF3 = 1;  // RF3 (SW0) configured as input

TRISFbits.TRISF5 = 1;  // RF5 (SW1) configured as input

TRISFbits.TRISF4 = 1;  // RF4 (SW2) configured as input

TRISDbits.TRISD15 = 1; // RD15 (SW3) configured as input

TRISDbits.TRISD14 = 1; // RD14 (SW4) configured as input

TRISBbits.TRISB11 = 1; // RB11 (SW5) configured as input

ANSELBbits.ANSB11 = 0;  // RB11 (SW5) disabled analog

TRISBbits.TRISB10 = 1; // RB10 (SW6) configured as input

ANSELBbits.ANSB10 = 0; // RB10 (SW6) disabled analog
```

```
TRISBbits.TRISB9 = 1;    // RB9 (SW7) configured as input

ANSELBbits.ANSB9 = 0;    // RB9 (SW7) disabled analog
```

4.2 Functionality

In order to read the switches, the user needs to read the corresponding digital input pin. A value of 1 indicates the switch as being on (high) or 0 indicates the switch as being off (low).

```
val = PORTFbits.RF3;  // read SW0

val = PORTFbits.RF5;  // read SW1

val = PORTFbits.RF4;  // read SW2

val = PORTDbits.RD15; // read SW3

val = PORTDbits.RD14; // read SW4

val = PORTBbits.RB11; // read SW5

val = PORTBbits.RB10; // read SW6

val = PORTBbits.RB9;  // read SW7
```

Library functions for using the switches are contained in the Basys MX3 library pack, SWT library; however, the user can easily use the switches without the SWT library, as presented above.

4.3 Shared Pins

As shown in the connectivity table above, SW7 driving signal is shared with the TRIG_2 signal in 2x15 pins Debug Header.

5 User Buttons

There are five buttons on the board, labeled BTNU, BTNL, BTNC, BTNR, BTND both on the board and in the schematic, attached to five digital I/O pins of PIC32. Reading the buttons is done by basic access to an input I/O pin. Read more details in Digital Inputs and Outputs section.

Figure 5.1 shows the way the buttons are electrically connected on the Basys MX3.

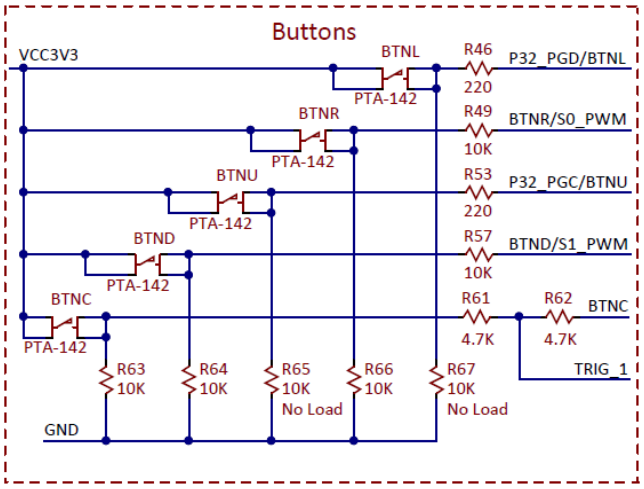


Figure 5.1. Button schematic diagram.

The Basys MX3 also has a red button labeled RESET. This button is connected directly to the MCLR pin of the PIC32 and will trigger it to be reset.

5.1 Connectivity

Label	Schematic Name	PIC32 Pin	Pin Shared With	Description
BTNU	BTNU	PGEC1/AN1/RPB1/CTED12/RB1	PGC	Button up
BTNL	BTNL	PGED1/AN0/RPB0/RB0	PGD	Button left
BTNC	BTNC	RPF0/PMD11/RF0	TRIG_1	Button center
BTNR	BTNR	AN8/RPB8/CTED10/RB8	S0_PWM	Button right
BTND	BTND	RPA15/RA15	S1_PWM	Button down

Table 5.1. Button connectivity.

All the pins must be defined as digital input: their corresponding TRIS bit must be set to 1, and analog function must be disabled for pins corresponding to BTNU, BTNL, BTNR, BTND.

```
TRISBbits.TRISB1 = 1; // RB1 (BTNU) configured as input

ANSELBbits.ANSB1 = 0; // RB1 (BTNU) disabled analog

TRISBbits.TRISB0 = 1; // RB1 (BTNL) configured as input

ANSELBbits.ANSB0 = 0; // RB1 (BTNL) disabled analog

TRISFbits.TRISF4 = 1; // RF0 (BTNC) configured as input

TRISBbits.TRISB8 = 1; // RB8 (BTNR) configured as input

ANSELBbits.ANSB8 = 0;   // RB8 (BTNR) disabled analog

TRISAbits.TRISA15 = 1;  // RA15 (BTND) configured as input
```

5.2 Functionality

To read the buttons, the user needs to read the corresponding digital input pin, a value of 1 indicating the button is pressed or 0 indicating the button is released:

```
val = PORTBbits.RB1;          // read BTNU

val = PORTBbits.RB0;          // read BTNL

val = PORTFbits.RF0;          // read BTNC

val = PORTBbits.RB8;          // read BTNR

val = PORTAbits.RA15;         // read BTND
```

Please note that if you want the buttons to trigger a specific functionality, proper software debouncing is required.

Library functions for using the buttons are contained in the Basys MX3 library pack, BTN library; however, the user can easily use the buttons without the BTN library, as presented above.

5.3 Shared Pins

As shown in the connectivity Table 5.1 above, some pins are shared:

- Buttons BTNL and BTNU share functions with PGD and PGC signals used for programming. Therefore, the following line should be inserted in the code, to disable their programming function.

  ```
  #pragma config JTAGEN = OFF
  ```

- Buttons BTNR and BTND share the pins with S0_PWM and S1_PWM, explained in Servo headers section, so these resources should be used exclusively.
- BTNC is shared with TRIG_1 signal in 2x15 Pins Debug Header, so it can be used to trigger events in an Analog Discovery board experiment.

6 RGB LED

The Basys MX3 board contains one tri-color (RGB) LED. The LED allows the user to obtain any RGB color by configuring the R, G and B color components.

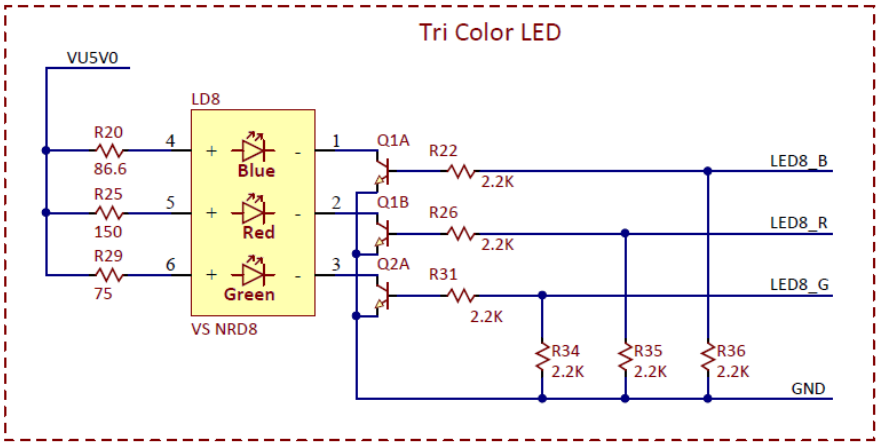


Figure 6.1. RGB LED schematic diagram.

The usage of the RGB LED is the same as controlling three separate LEDs, one for each color. Figure 6.1 shows the way the RGB LED is electrically connected on the Basys MX3.

There is one digital signal to control each color component. Using either 0 or 1 values for these signals will only give the user a limited number of colors (two colors for each component), so most of the time this is not enough in applications using the RGB feature. The solution is to send a sequence of 1 and 0 values on these digital lines, switched rapidly with a frequency higher than human perception. The "duty factor" will finally determine the color, as the human eye will "integrate" the discrete illumination values into the final color sensation.

The most used approach in solving this problem is the use of pulse-width modulation (PWM) signals. Another approach is the use of pulse-density modulation (PDM). These methods are explained in the RGB LED Implemented Using PWM and RGB LED Implemented Using PDM sections.

6.1 Connectivity

Label	Schematic Name	PIC32 Pin	Description
R	LED8_R	AN25/RPD2/RD2	Signal corresponding to the R component of the RGB
G	LED8_G	RPD12/PMD12/RD12	Signal corresponding to the G component of the RGB
B	LED8_B	AN26/RPD3/RD3	Signal corresponding to the R component of the RGB

Table 6.1. RGB LED connectivity.

6.2 Functionality

6.2.1 RGB LED Implemented Using PWM

The percentage of each period that the pulse is high determines the signals "duty cycle". Figure 6.2 shows how different duty cycles are implemented using PWM.

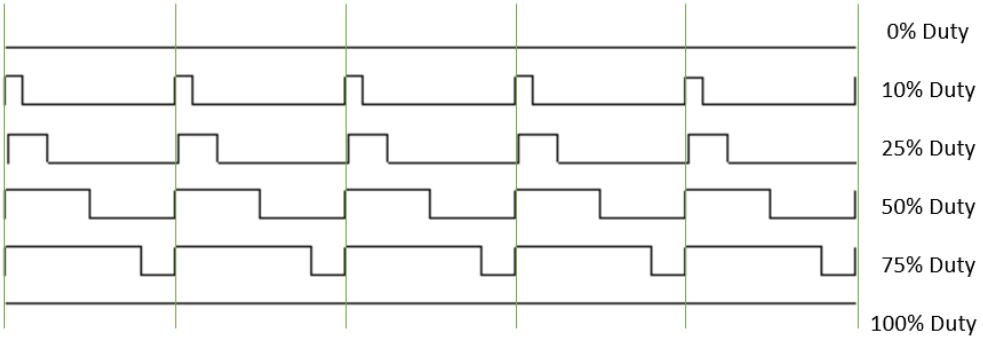


Figure 6.2. PWM duty cycle.

Using this method, the intensity of each component of the RGB LED is determined by the duty cycle being applied. PWM is most often implemented in the microcontroller using the output compare (OC) peripheral modules along with a timer.

One timer (Timer y) is assigned to the OC module. Setting the PRy register of the timer will set the PWM period. Setting the OCxRS register of the OC module will set the actual duty cycle.

The PIC32 datasheet displayed in Fig. 6.3 below explains how one period of the PWM is generated.

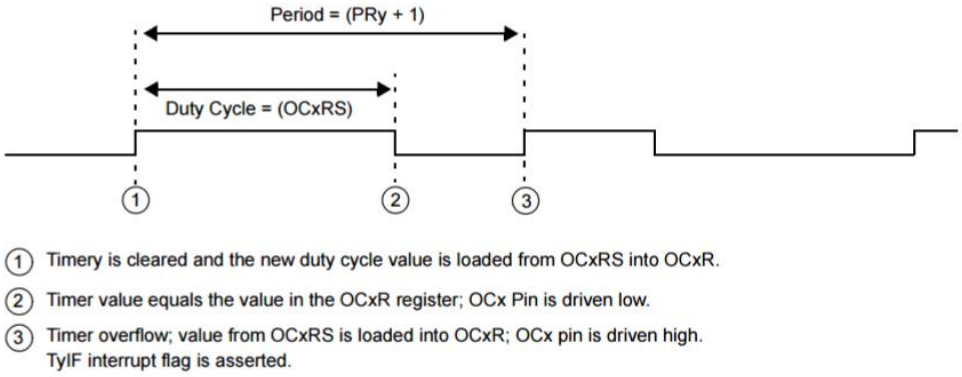


Figure 6.3. PIC32 PWM generation.

The RGBLED library from the Basys MX3 library pack contains a commented example of PWM implementation with the following features:

- LED8_R, LED8_G, and LED8_B are mapped to OC3, OC5, and OC4.
- OC3, OC5, and OC4 are properly configured, together with assigned Timer 2.
- When a new color is set, its components (R, G, and B) are assigned to OC3RS, OC5RS, and OC4RS.

6.2.2 RGB LED Implemented Using PDM

PDM method adjusts both the frequency and length of the "High" pulses in the modulated signal.

A PDM is implemented using a register and an accumulator adder with carry output. The n-bit register can store any binary value from 0 to $2^n - 1$. In each clock period, the register content is added to the accumulator. The carry bit (overflow of the n-bit accumulator) is the output. It is "High" as often as the accumulator overflows, so that when large values are added, carry will occur often. The "High" pulse is only 1 clock period long, but more "High" pulses can succeed when the register content is close to maximum.

The RGBLED library from the Basys MX3 library pack contains an example of PDM implementation, with the following features:

- LED8_R, LED8_G, and LED8_B are configured as simple digital outputs.
- Timer 5 is configured to generate an interrupt every approx. x us.
- Three 16-bit accumulators are used, one for each color.
- In the interrupt service routine, for each color, the 8-bit color value is added to the corresponding 16-bit accumulator.
- For each color, the 9th bit of the accumulator is considered the carry bit. The resulted carry bits are assigned to LED8_R, LED8_G, and LED8_B.
- For each color, the accumulator is masked so that it only contains an 8-bit value (carry is cleared).

7 Seven-segment Display

The Basys MX3 board contains a four-digit common anode seven-segment LED display. Each of the four digits is composed of seven segments displaying a "figure 8" pattern and a decimal point with an LED embedded in each segment. Segment LEDs can be individually illuminated. Of the number of possible patterns, the ten corresponding to the decimal digits are the most useful.

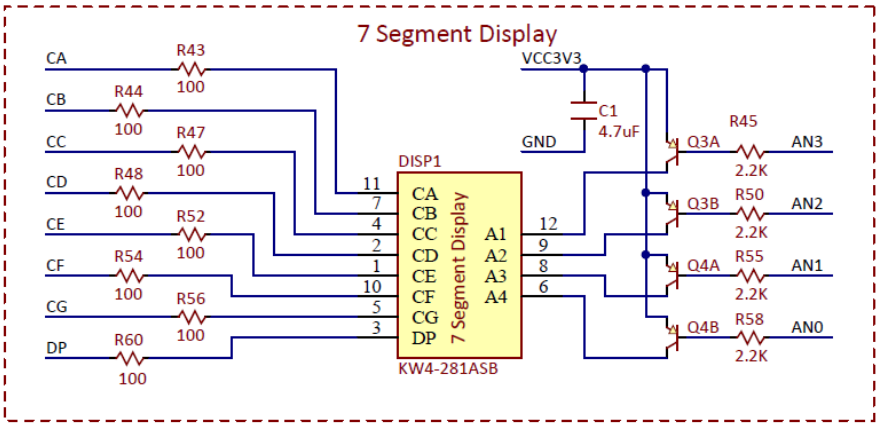


Figure 7.1. Seven-segment schematic diagram.

The anodes of the seven LEDs forming each digit are tied together into one "common anode" circuit node, but the LED cathodes remain separate, as shown in Fig. 7.2. The common anode signals are available as four "digit enable" input signals to the 4-digit display. The cathodes of similar segments on all four displays are connected into seven circuit nodes labeled CA through CG (so, for example, the four "D" cathodes from the four digits are grouped together into a single circuit node called "CD"). These seven cathode signals are available as inputs to the 4-digit display. This signal connection scheme creates a multiplexed display, where the cathode signals are common to all digits but they can only illuminate the segments of the digit whose corresponding anode signal is asserted.

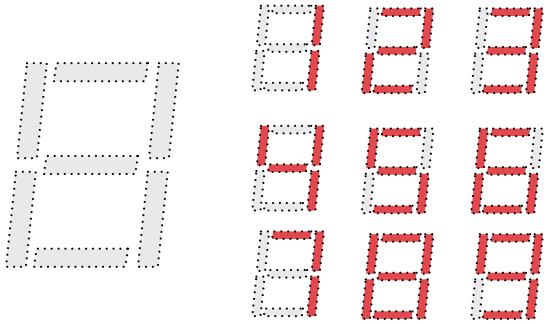


Figure 7.2. Seven-segment digits.

To illuminate a segment, the anode should be driven high while the cathode is driven low; however, since the Basys MX3 uses transistors to drive enough current into the common anode point, the anode enables are inverted. Therefore, both the AN0 … AN3 and the CA … G/DP signals are driven low when active.

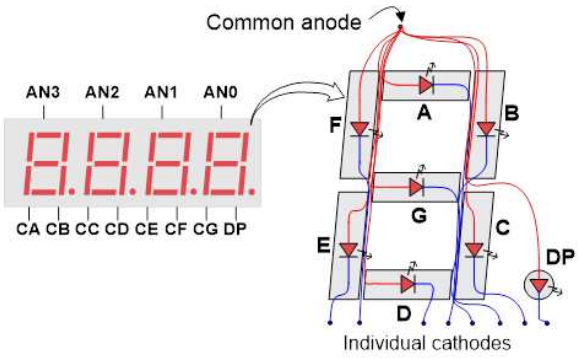


Figure 7.3. Common anode circuit node.

A scanning display controller circuit can be used to show a 4-digit number on this display. This circuit drives the anode signals and corresponding cathode patterns of each digit in a repeating, continuous succession at an update rate that is faster than the human eye can detect. Each digit is illuminated just one-fourth of the time, but because the eye cannot perceive the darkening of a digit before it is illuminated again, the digit appears continuously illuminated. If the update, or "refresh", rate is slowed to around 45Hz, a flicker can be noticed on the display.

To make each of the four digits appear bright and continuously illuminated, all should be driven once every 1 to 16ms, with a refresh frequency of about 1kHz to 60Hz. For example, if every digit is refreshed every 3ms, corresponding to a frequency of 333Hz, the entire display will be refreshed every 12ms. The controller must drive low the cathodes with the correct pattern when the corresponding anode signal is driven high.

To illustrate the process:

1. If AN0 is asserted while CB and CC are asserted, then a "1" will be displayed in digit position 1.
2. If AN1 is asserted while CA, CB, and CC are asserted, a "7" will be displayed in digit position 2.
3. If AN0, CB, and CC are driven for 4ms, and then AN1, CA, CB, and CC are driven for 4ms in an endless succession, the display will show "71" in the first two digits. An example timing diagram for a four-digit controller is shown in Fig. 7.4.

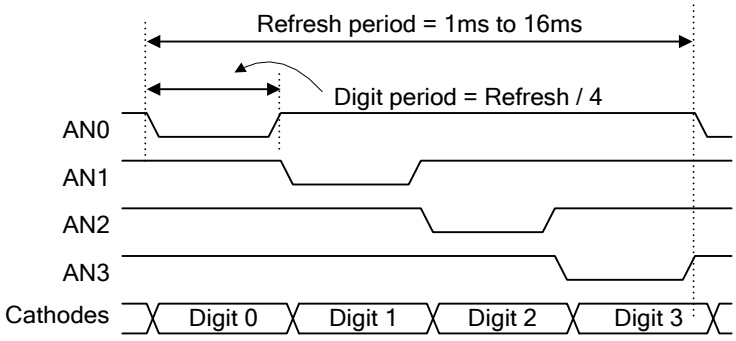


Figure 7.4. 4-digit scanning display controller timing diagram.

Please note that the two dots (situated between the middle digits) are not connected.

7.1 Connectivity

Name	PIC32 Pin	Description
AN0	AN12/PMA11/RB12	Anode 0 pin
AN1	AN13/PMA10/RB13	Anode 1 pin
AN2	VREF-/CVREF-/PMA7/RA9	Anode 2 pin
AN3	VREF+/CVREF+/PMA6/RA10	Anode 3 pin
CA	TRD1/RG12	Cathode A pin
CB	RPA14/RA14	Cathode B pin
CC	PMD14/RD6	Cathode C pin
CD	TRD0/RG13	Cathode D pin
CE	RG15	Cathode E pin
CF	PMD15/RD7	Cathode F pin
CG	PMD13/RD13	Cathode G pin
DP	TRD2/RG14	Cathode DP (decimal point) pin

Table 7.1. Seven-segment connectivity.

All the pins (AN0-3, CA-CG and DP) must be configured as digital output and Anode 0 and Anode 1 must have the analog functionality disabled:

```
TRISBbits.TRISB12 = 0; //RB12 set as output

ANSELBbits.ANSB12 = 0; //RB12 analog functionality disabled

TRISBbits.TRISB13 = 0; //RB13 set as output

ANSELBbits.ANSB13 = 0; //RB13 analog functionality disabled

TRISAbits.TRISA9 = 0; //RA9 set as output

TRISAbits.TRISA10 = 0; //RA10 set as output

TRISGbits.TRISG12 = 0; //RG12 set as output

TRISAbits.TRISA14 = 0; //RA14 set as output

TRISDbits.TRISD6 = 0; //RD6 set as output

TRISGbits.TRISG13 = 0; //RG13 set as output

TRISGbits.TRISG15 = 0; //RG15 set as output

TRISDbits.TRISD7 = 0; //RD7 set as output

TRISDbits.TRISD13 = 0; //RD13 set as output

TRISGbits.TRISG14 = 0; //RG14 set as output
```

7.2 Functionality

A seven-segment display controller is implemented in the SSD library of the Basys MX3 library pack. Here are some details on the implementation of the library:

- One array contains constant values for the segment's configurations (one bit for each segment) corresponding to various digits (0-9, A-F).
- When the user selects the values to be displayed, they are used as index into this segment's configuration table and the resulting configuration bytes are stored in global variables.
- Timer1 is used to generate interrupts every 3ms (corresponding to the period register PR1 = 3750).
- Every time the interrupt handler routine is called, the following operations are performed:
    - The next digit becomes the current digit, in a circular approach (thus each digit will be addressed once after 4 calls of the interrupt handler routine).
    - All digits are deactivated by outputting 1 to their corresponding anodes.
    - The cathodes are outputted according to the segment's information corresponding to the current digit.
    - The current digit is activated (0 is outputted to its corresponding anode).

8 LCD Module

The Basys MX3 features a basic LCD module, the Sunlike Display SD1602H with a KS0066U display controller. It displays two rows of 16 characters. It is controlled using a set of command signals (DISP_RS, DISP_R/W, DISP_EN) and 8 data signals (DB0 - DB7). These signals make up a parallel port for communicating with the display.

The board also provides a switch to turn on and off the LCD display backlight, situated on the bottom right corner of the LCD display.

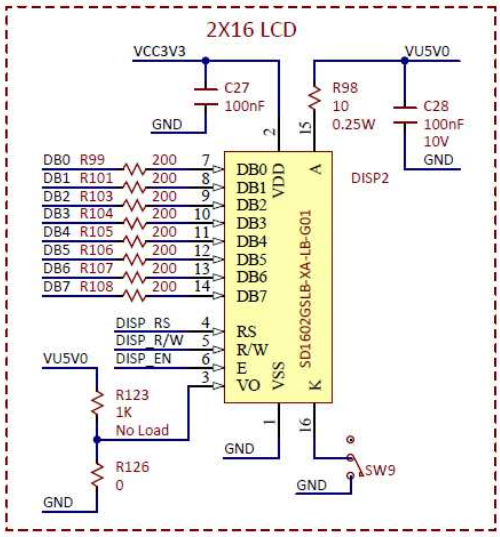


Figure 8.1. LCD schematic diagram.

The LCD display is controlled by a set of commands written to the device. Also, read commands provide the ability to read status and data. Please read the SD1602H datasheet for a detailed list of read and write commands. The LCD display requires a specific initialization sequence, also detailed in the datasheet.

The device features two types of memory: CGRAM and DDRAM.

The LCD controller contains a character-generator ROM (CGROM) with 192 preset 5×8 character patterns, a character-generator RAM (CGRAM) that can hold 8 user-defined 5×8 characters, and a display data RAM (DDRAM) that can hold 80 character codes. Character codes written into the DDRAM serve as indexes into the CGROM (or CGRAM). Writing a character code into a particular DDRAM location will cause the associated character to appear at the corresponding display location. Display positions can be shifted left or right by setting a bit in the instruction register (IR). The write-only IR directs display operations (such as clear display, shift left or right, set DDRAM address, etc.). A busy flag shows whether the display has completed the last requested operation; prior to initiating a new operation, the flag can be checked to see if the previous operation has been completed.

The display has more DDRAM locations than can be displayed at any given time. DDRAM locations 00H to 27H map to the first display row, and locations 40H to 67H map to the second row. Normally, DDRAM location 00H maps to the upper left display corner (the “home” location), and 40H to the lower left. Shifting the display left or right can change this mapping. The display uses a temporary data register (DR) to hold data during DDRAM /CGRAM reads or writes, and an internal address register to select the RAM location. Address register contents, set via the instruction register, are automatically incremented after each read or write operation. The LCD display uses ASCII character codes. Codes up through 7F are standard ASCII (which includes all “normal” alphanumeric characters). Codes above 7F produce various international characters.

The following timing diagrams detail how write and read processes must be implemented. The essential difference is the polarity of the DISP_RW signal (0 for write and 1 for read). For more detailed timing information, refer to the KS0066U datasheet.

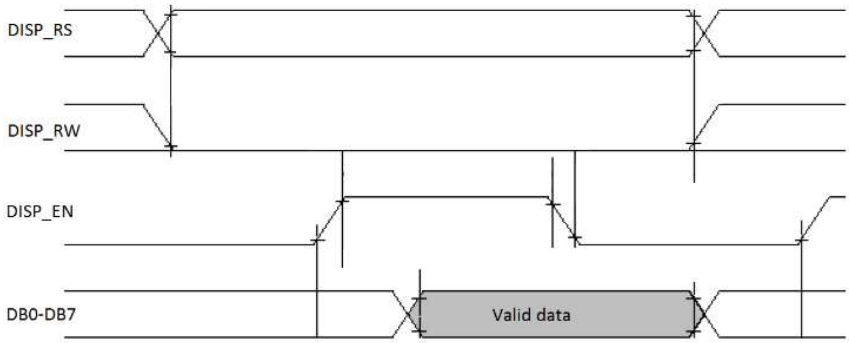


Figure 8.2. LCD write timing.

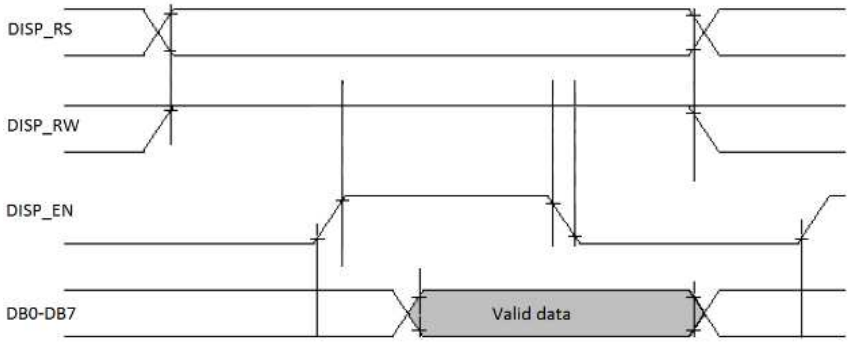


Figure 8.3. LCD read timing.

8.1 Connectivity

Name	PIC32 Pin	Description
DISP_RS	AN15/RPB15/OCFB/CTED6/PMA0/RB15	Register Select: High for Data Transfer, Low for Instruction Transfer.
DISP_RW	RPD5/PMRD/RD5	Read/Write signal: High for Read mode, Low for Write mode.
DISP_EN	RPD4/PMWR/RD4	Read/Write Enable: High for Read, falling edge writes data
DB0	PMD0/RE0	Data bits 0 -7.
DB1	PMD1/RE1	
DB2	AN20/PMD2/RE2	
DB3	RPE3/CTPLS/PMD3/RE3	
DB4	AN21/PMD4/RE4	
DB5	AN22/RPE5/PMD5/RE5	
DB6	AN23/PMD6/RE6	
DB7	AN27/PMD7/RE7	

Table 8.1. LCD connectivity.