Lab 2: Interfacing with a Sensor Device on an Embedded Computer System

16.480/552 Microprocessor Design II and Embedded Systems

Instructor: *Prof. Yan Luo*

Group-I

Due:11/07/16

Submitted: 11/07/16

By,

- ***Naga Ganesh Kurapati***
- *Sayali Vaidya*
- *Zubin Pattnaik*
- *Akriti Sharan*

Table of contents

I. Contributions
II. Purpose
III. Introduction
IV. Materials, Devices and Instruments Used
V. Schematics
VI. Lab Methods and Procedure
VII. Trouble Shooting
VIII. Results
IX. Appendix - Code

I. Contributions

Naga Ganesh Kurapati – Worked on configuring the microcontroller PIC16F18857 using MPLAB and setting up the circuit. Assisted to synchronize the strobe communication between the PIC and Galileo. Debugging the codes.

Zubin Patnaik – Worked on designing the pic circuit, troubleshooting the ports and circuit. Debugging the codes.

Sayali Vaidya – Worked on configuring the ports of Galileo for the lab. Assisted to synchronize the strobe communication. Debugging the codes.

Akriti Sharan – Worked on setting up the Linux on Galileo board, assisted in troubleshooting the circuit and synchronize the strobe communication.

II. Purpose

The main purpose of this project is to design bus protocol between intel Galileo (master) and PIC (slave). And transmit the ADC value read by PIC to the Galileo board over bus protocol using strobe mechanism. And display those value on the computer screen using putty software.

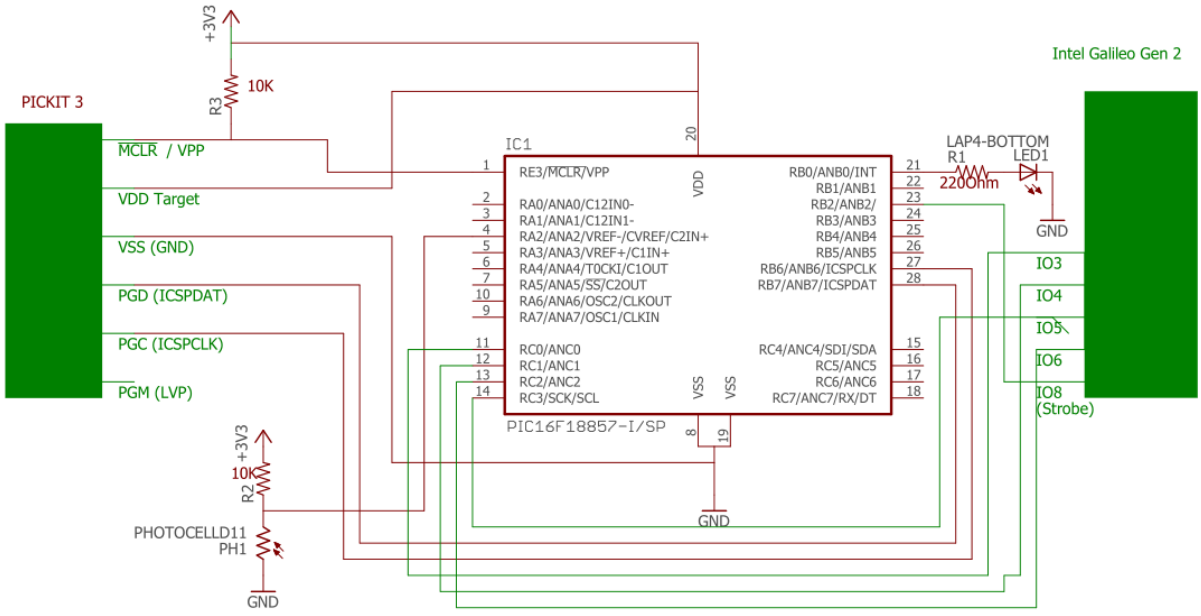
III. Introduction

The main objective of this lab is to read the data from a photo resistor through ADC module of the microcontroller PIC16F18857. Send those 10-bit data, over the 4-bit bus using a strobe mechanism, by breaking the value to 4 bits. That data is read by Intel Galileo board from it GPIO ports. It reassembles the data into 10 bits and display the decimal value on the screen. Strobe is controlled by Galileo Board. It reads or writes the data to ports keeping strobe high.

IV. Materials, Devices and Instruments Used

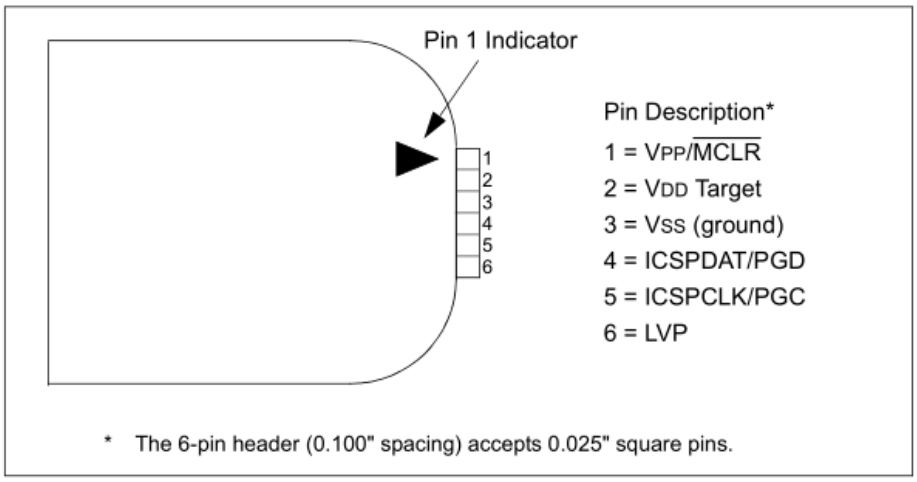
1. Bread board
2. Wires to connect
3. PIC16F18857 microcontroller
4. Pickit3
5. LED
6. Photo resistor
7. two 10k, one 220 Ohm resistors
8. Serial to USB connector
9. Multi-meter
10. Voltage supply (3.3V)
11. Intel Galileo Gen 2 Board
12. Yocto Linux
13. MPLAB IDE
14. Putty Software

V. Schematics



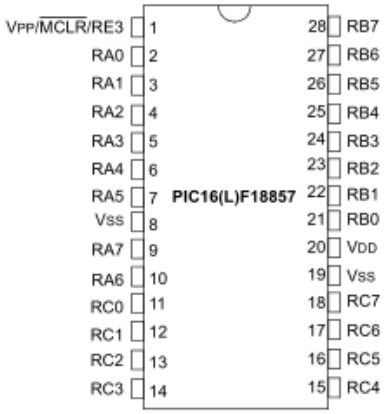
VI. Lab Methods and Procedure

FIGURE 1-2: PICKIT™ 3 PROGRAMMER CONNECTOR PINOUT



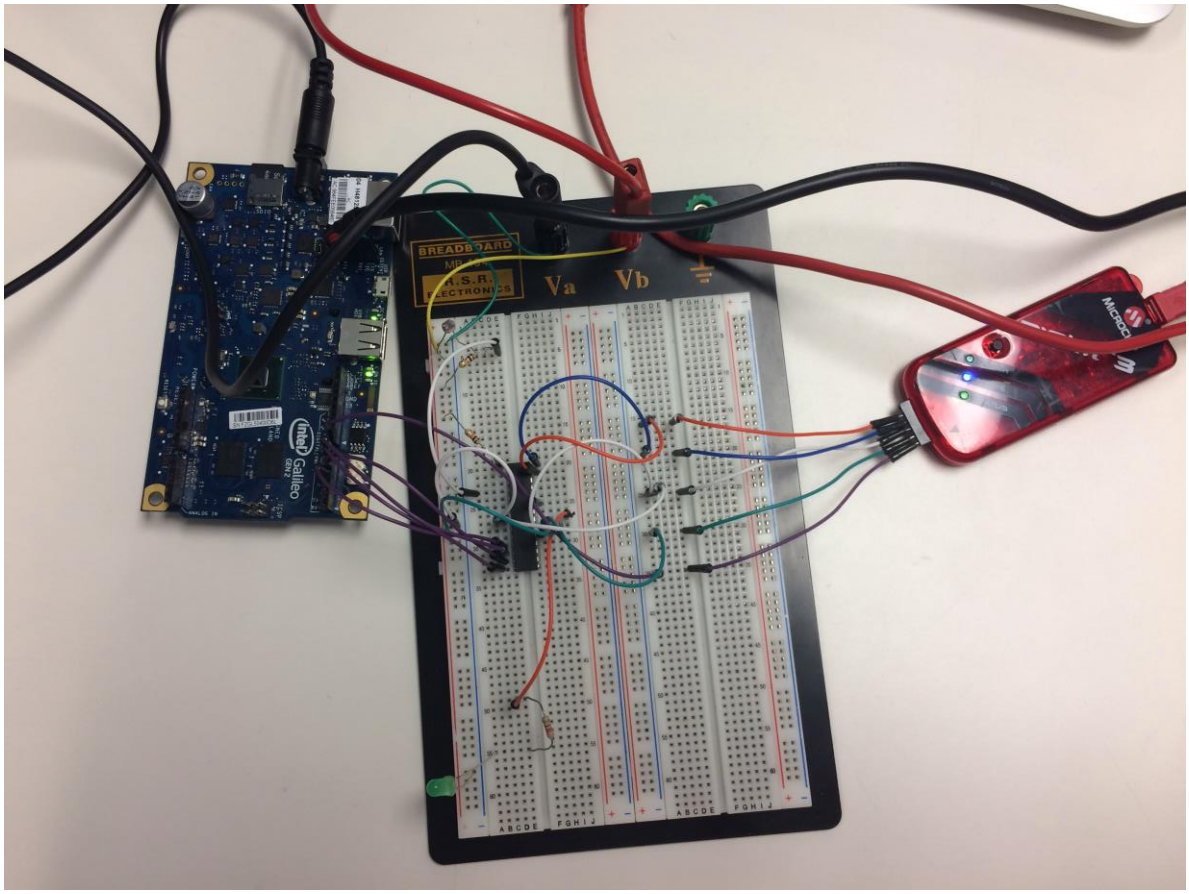
PIN DIAGRAMS

28-pin SPDIP, SOIC, SSOP



Note 1: See Table 2 for location of all peripheral functions.

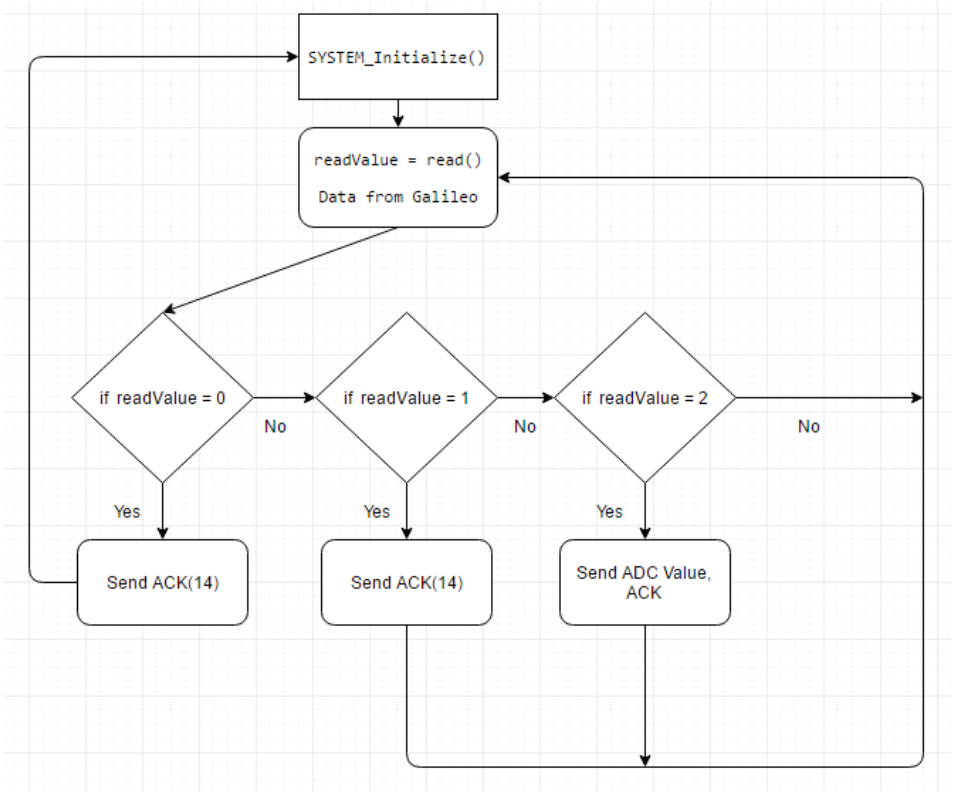
2: All VDD and all VSS pins must be connected at the circuit board level. Allowing one or more VSS or VDD pins to float may result in degraded electrical performance or non-functionality.



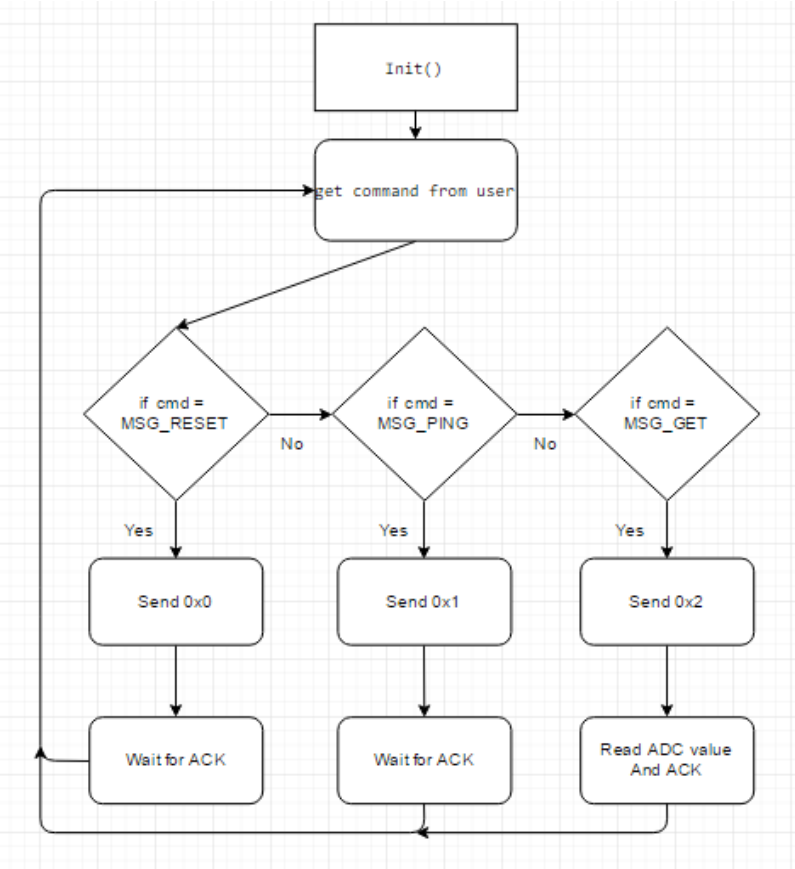
Hardware Design: Initially Pickit3 is connected to the microcontroller. If you observe the pin diagram of both Pickit 3 on top and PIC. Both MCLR, Vdd, Vss, ICSPDAT/PGD, ICSPCLK/PGC are connected to each other. ICSPDAT is pin 27 and ICSPCLK is pin 28 for the PIC. The MCLR is connected to Vdd through 10K ohm resistor. The sensor is connected through ADC Channel 2(Pin 4). And LED is connected to the pin PB0 (Pin21). A 220-ohm resistor is connected in series to the LED, for protection. Pin RB2 is connected to strobe(GPIO8) of Galileo. RC0, RC1, RC2 & RC3 pins are connected to the GPIO3,4,5,6 pins of Intel Galileo.

Software Design:

PIC flow chart



Galileo Flowchart



If you see the both flow chart, they look similar in operation. They work like state machine. The GPIO ports of Galileo is configured using shell script found in Appendix. PIC pins are configured using MPLAB code configure. The main skeleton of the code operating is as shown in the flowcharts. The communication between the two devices is explained in result section.

VII. Trouble Shooting

The first is to assure the correctness of the circuit by blinking the LED for few minutes without the sensor data. Then relate the sensor data to ON the led. To check the sensor, you need to measure the voltage across it during blocked and unblocked situations. The measured values are 2.8v during unblocked and 1v during the blocked. To estimate ADC output using the formula. Signal = (sample/1024) * Reference voltage. For 2.8v, sample = 868. For 1v, sample = 310 with reference voltage = 3.3 v and (2^10- 10bits of ADC value) 1024 base value.

The second is to troubleshot the PORTC- data pins configurations by blinking the LED at every pin used.

Third is to configure GPIO ports 3,4,5,6,8 of Intel Galileo by blinking LED at each port used. PORT configuration code can be found in the Appendix.

VIII. Results

```
root@clanton:~# ls
LAB_2.c  intel_new.c               pin13.sh.txt    pin_out.sh
a.out    lab2                      pin4_output.sh  pin_out_strobe.sh
error    lab2_galileo_firstCopy.c  pin5.sh         test
intel    out.txt                   pin5_output.sh  test.c
intel.c  pin13.sh                  pin_in.sh
root@clanton:~# gcc -o intel intel.c
root@clanton:~# rm intel
root@clanton:~# gcc -o intel intel.c
root@clanton:~# ./intel
Enter command
1.MSG_RESET 2.MSG_PING 3. MSG_GET)MSG_RESET
You have entered MSG_RESET
Send value: 0
Received Value:14
Enter command
1.MSG_RESET 2.MSG_PING 3. MSG_GET)MSG_PING
You have entered MSG_PING
Send value: 1
Received Value:14
Enter command
1.MSG_RESET 2.MSG_PING 3. MSG_GET)MSG_GET
You have entered MSG_GET
Send value: 2
Nibl1:14
Nibl2:14
Nibl3:14
Ack:14
ADC Result:750
```

From the above fig., you can see the communication that is happening between the PIC and intel Galileo. MSG_RESET(0x0) is the message send to PIC from Galileo, in response it resets the PIC ports and PIC sends ACK(0x14) to Galileo. If MSG_PING is the message send to PIC from Galileo, in response PIC sends ACK(0x14) to Galileo. If MSG_GET is the message send to PIC from Galileo, in response PIC sends 10-bit ADC value in form of 3 consecutive 4-bit data and followed by ACK(0x14) to Galileo.

IX. Appendix – Code

1. To configure Intel Galileo GPIO pins as input

```
---------------------------------------------------------
#!/bin/sh
echo -n "$1" > /sys/class/gpio/export
echo -n "in" > /sys/class/gpio/gpio$1/direction
echo -n "$2" > /sys/class/gpio/export 2>>error
echo -n "in" > /sys/class/gpio/gpio$2/direction
cat  /sys/class/gpio/gpio$2/value 1> out.txt
echo -n "$1" > /sys/class/gpio/unexport
echo -n "$2" > /sys/class/gpio/unexport
```

2. To configure Intel Galileo GPIO pins as output

```
---------------------------------------------------------
#!/bin/sh
echo -n "$1" > /sys/class/gpio/export
echo -n "out" > /sys/class/gpio/gpio$1/direction
echo -n "$2" > /sys/class/gpio/export 2>>error
echo -n "out" > /sys/class/gpio/gpio$2/direction
echo -n "$3" > /sys/class/gpio/gpio$2/value
echo -n "$2" > /sys/class/gpio/unexport
echo -n "$1" > /sys/class/gpio/unexport
```

3. Read function in PIC

```
-----------------------------------------
int read(void)
{
  int data;
  IO_RC0_SetDigitalInput(); // To set RC0 as Input
  IO_RC1_SetDigitalInput(); // To set RC1 as Input
  IO_RC2_SetDigitalInput(); // To set RC2 as Input
  IO_RC3_SetDigitalInput(); // To set RC3 as Input
  while(!IO_RB2_GetValue()); // Wait for strobe high
  while(IO_RB2_GetValue()) // During strobe high, write the data to
  data = PORTC;                                                  ports
  return data;
}
```

4. Write function in PIC

```
-----------------------------------------
void write(int data)
{
 IO_RC0_SetDigitalOutput(); // To set RC0 as Output
 IO_RC1_SetDigitalOutput(); // To set RC1 as Output
 IO_RC2_SetDigitalOutput(); // To set RC2 as Output
 IO_RC3_SetDigitalOutput(); // To set RC3 as Output
 while(!IO_RB2_GetValue()); // Wait for strobe high
 while(IO_RB2_GetValue())// During strobe high, read the data from
 PORTC = data;                                                port
 PORTC = 0x0;
}
```

5. Main function in PIC

---

void main(void)

```
{
 SYSTEM_Initialize(); // initialize ports, configure them
 while (1)
 {
     int readValue = read(); // read the value from port
     if(readValue==0x0) // If MSG_RESET,call system initialize&Send ACK
     {
         SYSTEM_Initialize();
         write(0xE); //ACK
     }
     else if(readValue==0x1) //If MSG_PING, send ACK
     {
         write(0xE); //ACK
     }
     else if(readValue==0x2)// If MSG_GET, send ADC value and ACK
     {
         uint16_t adc_result = ADCC_GetSingleConversion(channel_ANA2);
         unsigned int nib1, nib2, nib3; // Extract nibble 1,2,3
         nib1 = adc_result & 0x0f;
         nib2 = (adc_result >> 4) & 0x0f;
         nib2 = (adc_result >> 8) & 0x03;
         write(nib1);
         write(nib2);
         write(nib3);
         write(0xE); //ACK
     }
 }
}
```