EMBEDDED SYSTEMS

Assignment

Master of Science Robotics Engineering

Group
Ankur Kohli (5160903)
Ammar Iqbal (5183355)
Basit Akram (5161322)

Submitted to
Prof. ENRICO SIMETTI

Dipartimento di Informatica, Bioingegneria, Robotica e
Ingegneria dei Sistemi (DIBRIS)
Universit'a degli Studi di Genova (UniGe)

Preface

The work is based on Embedded Systems which involves the implementation of timers, interrupts, SPI, UART, parser ADC, PWM, Scheduling, and so on. This assignment deals with MPLAB IDE Software, XC16 Compiler, HTerm serial software and Microchip Microcontroller Board.

Acknowledgements

We take this opportunity to express our sincere thanks to *Prof. ENRICO SIMETTI* for his guidance and support. We would also like to express our gratitude towards him for showing confidence in us. It was a privilege to have a great experience working under him in a cordial environment. We are very much thankful to the *University of Genoa*, for providing us the opportunity to pursue *M.Sc in Robotics Engineering* in a peaceful environment with ample resources. In the end, we would like to acknowledge our parents, friends. Without their support, this work would not have been possible.

Abstract

This assignemnt report is about the ***Embedded Systems*** *in which the of* ***timers, interrupts, SPI, UART, parser, ADC, PWM, Scheduling, and so on*** *to determine about the operations performance. The specific goal in this case is that to know the implementation of operations with the real-time hardware. Also,* ***case study of a real time control system****. This report to bring the light on* ***Embedded Systems*** *for* ***operations****. Furthermore, the purpose of this report is to provide the approaches used during the development of code and implementation on microcontroller board. For this assignment,* ***MPLAB IDE Software, XC16 Compiler and Microchip Microcontroller Board, HTerm serial software*** *is used and also Embedded C programming is platform for the development of code.*

Contents

1 Introduction

A microprocessor-based computer system with software that is intended to carry out a specific task, either independently or as a component of a larger system, is known as an ***embedded system***. An integrated circuit built to perform computing for real-time processes is at the heart of the system.

From a single microcontroller to a group of connected processors with networks and peripherals, complexity can range from having no user interface to having intricate graphical user interfaces. Depending on the task for which it is created, an embedded system's complexity varies greatly.

Applications for embedded systems include hybrid cars, avionics, digital watches, microwaves, and more. Embedded systems consume up to 98 percent of all produced microprocessors.

2 Methodlogy

In this assignment, our main task is to implement the operations such as ***timers, interrupts, SPI, UART, parser, ADC, PWM, Scheduling, and so on***. Also, this assignment deals with MPLAB IDE Software, XC16 Compiler, HTerm serial software and Microchip Microcontroller Board. We used ***dsPIC30F4011 Enhanced Flash 16-bit Digital Signal Microcontroller Board*** as shown in figure 1, for the real-time implementation of the developed code.

Peripheral Features of the board:

- High current sink/source I/O pins: 25 mA/25 mA
- Timer module with programmable prescaler: Five 16-bit timers/counters; optionally pair 16-bit timers into 32-bit timer modules
- 16-bit Capture input functions
- 16-bit Compare/PWM output functions
- 3-wire SPI™ modules (supports 4 Frame modes)
- I2C™ module supports Multi-Master/Slave mode and 7-bit/10-bit addressing
- 2 UART modules with FIFO Buffers

- 1 CAN modules, 2.0B compliant

Source: dsPIC30F4011 Datasheet https://ww1.microchip.com/downloads/en/devicedoc/70135c.pdf

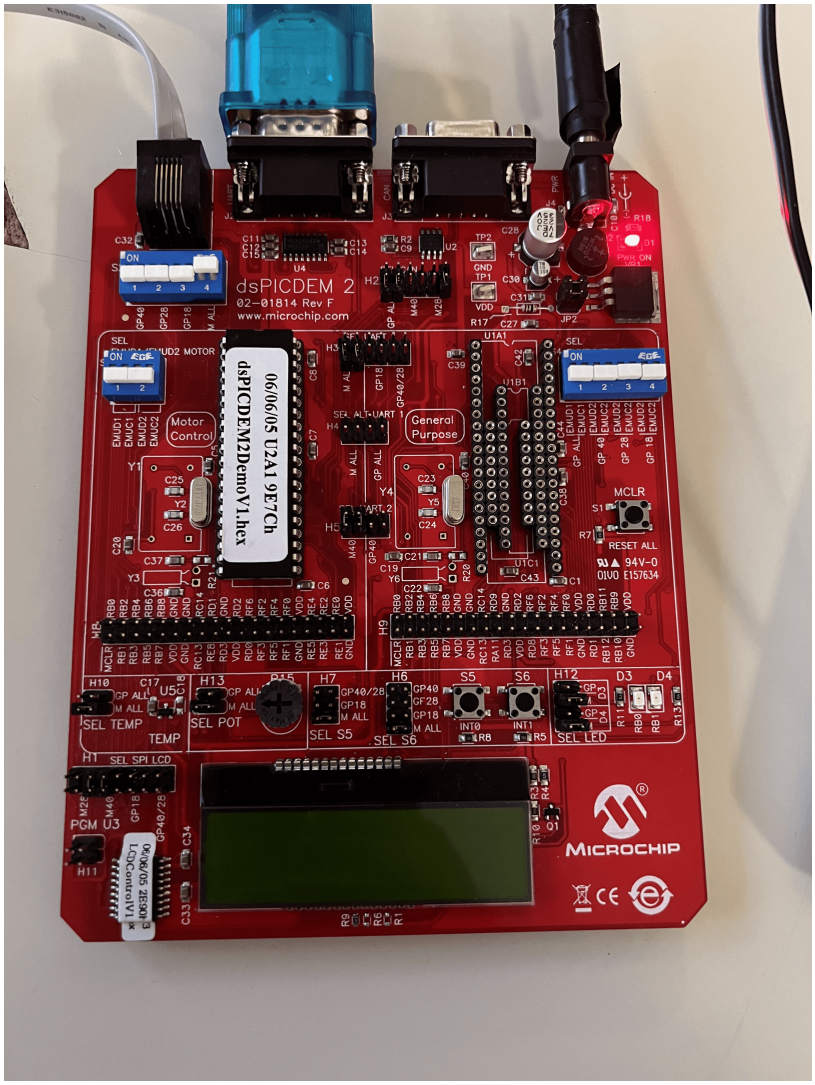

Figure 1: dsPIC30F4011 Board

DSP Engine Features:

- Dual data fetch
- Accumulator write back for DSP operations
- Modulo and Bit-Reversed Addressing modes
- Two, 40-bit wide accumulators with optional saturation logic
- 17-bit x 17-bit single cycle hardware fractional/ integer multiplier

- All DSP instructions single cycle
- ± 16-bit single cycle shift

Source: dsPIC30F4011 Datasheet https://ww1.microchip.com/downloads/en/devicedoc/70135c.pdf

Porgramm Burner: pickit3 Programmer

Figure 2, shows the program burner.

Figure 2: pickit3 Programmer

Requirements for the assignment

1. execute a control loop at 200 Hz.
2. the reference speed for the motor is received through UART communications. Setup baudrate at 9600 bps.
3. set the voltage to the armature of the DC motor using PWM2H. Assume that the motor spins at 1000 RPM at 5V.
4. simulate an analog current sensor using AN2 (potentiometer). The sensor outputs 3 V at 0 A, and has a scale of 10 A/V.
5. send the current and temperature (AN3) feedback through UART at 1 Hz.
6. Whenever button S6 is pressed, clear the first row and reset the characters received counter.
7. blink D3 at 1 Hz to show that the program is running correctly.
8. turn on D4 whenever the current exceeds 15A.

UART Protocol:

- The PC sends $MCREF,RPM*, where RPM is a value going from 0 to 1000. The references can come as fast as the baudrate allows. Example: $MCREF,400*
- The micro sends $MCFBK,CURRENT,TEMP*, where CURRENT is the current value in Amperes and TEMP is the temperature value at 1 Hz. Example: $MCFBK,4.3,22.2*

Tips for Writing to LCD with SPI

The following power-up sequence should be observed by the user's application firmware when writing characters to the LCD:

1. After any reset operation wait 1000 milliseconds to allow the LCD to begin normal operation. The cursor on the LCD will be positioned at the top row on the left-most column.
2. Configure SPI1 module on your dsPIC30F device to operate in 8-bit Master mode. The serial clock may be set for any frequency up to 1 MHz.
3. To write an ASCII character to the LCD at the location pointed to by the cursor, load the SPIBUF register with the ASCII character byte.
4. After the character is displayed on the LCD, the cursor is automatically relocated to the next position on the LCD.
5. To reposition the cursor to another column on any of the two rows, write the address of the desired location to the SPIBUF register. Addresses in the first row of the LCD range from 0x80 to 0x8F, while addresses on the second row range from 0xC0 through 0xCF.
6. After 16 characters are written to the first row on the LCD, it is necessary for the user's application to write the address 0xC0 of the second row to the SPIBUF in order to roll the cursor over to the second row.
7. The user application must wait for a minimum of (8 bits / SPI Frequency) between writing two successive characters or addresses.

Source: Assignemnt pdf https://2022.aulaweb.unige.it/mod/assign/view.php?id=70551

Scenario of the assignment

The scenario is divided into two task:

Scenario 1: This scenario is divided into two task.

Task 1 (PC to Controller): In this task Motor is control using frequency 200 Hz.

Task 2 (Controller to PC): In this task, monitoring of analog values at 1 Hz frequency.

Task 1:

- In this task, values were sent from PC (Personal Computer) to microcontroller board using UART.
- Motor is controlled at a frequency of 200 Hz.
- Moreover, receive value from UART will save in the circular buffer.
- After storing values, these values will be decoded by using ***parser*** *(parser.zip is already given during the task implementation).*
- Decoded value will be the RPM of the motor. This will generate PWM.

Task 2:

- In this task, values were received from microcontroller to PC. On microcontroller, there were enormous analog pins to receive analog data from pins such as temperature sensor, potentiometer, and so on.
- Analog values were converted into digital values using ADC which were received on PC.
- Analog input values are current, temperature, at 1 Hz frequency.

Scenario 2: There are three task as described below:

Task 1: In this task, reads a value from the potentiometer *AN2* pin. These value will range from 0 to 1023. Now, these values were mapped into voltage and current. For voltage we used map function as given below:

General map function:

$$output\_value = map(input\_value,\ input\_minimum\_value,\ input\_maximum\_value,\ output\_minimum\_value,\ output\_maximum\_value)$$

Map function in our case:

$$\boldsymbol{voltage = map\ (volt,\ 3,\ 5,\ 0,\ 20)}$$

After mapping into voltage, using these values and remap to get current values. For current values below are the ranges:

$$3V = 0A$$
$$4V = 10A$$
$$5V = 20A$$

where V = Voltage, A = Ampere

Map function for current values:

$$\boldsymbol{current = map\ (AN2,\ 0,\ 1023,\ 0,\ 5)}$$

Here, current = output value after map function, AN2 = input value from analog pin of the controller, 0 = input minimum value, 1023 = input maximum value, 0 = output minimum value, 5 = output maximum value.

Task 2: This task will read temperature value from *AN3* pin of controller board. This temperature value will ranges from 0 to 1023.

Below are the Temperature senor specifications:

- Temperature sensor, U5, is a -40°C to +125°C linear output TC1047A connected to analog channel AN3 of the dsPIC30F device through header H10.
- The output of the temperature sensor is fed directly to the dsPICDSC device.
- The output voltage range for the TC1047A is typically 750 mV at +25°C.
- The TC1047A exhibits a typical 10 mV/C voltage slope.

Source: https://2022.aulaweb.unige.it/pluginfile.php/267294/mod_resource/content/0/03_6_ADC.pdf

After receiving the temperature value from AN3 pin mapping these value from -40°C to +125°C in the form of voltage. For temperature values below are the ranges:

$$0.1V = -40^{\circ}C$$

$$0.5V = 0^{\circ}C$$

$$0.75V = 25^{\circ}C$$

$$1.75V = +125^{\circ}C$$

The aforementioned values were calculated by the dry run of the formula as given below:

$$Temp = AN3\ X\ Num\ -40$$

where, Temp = Output value, AN3 = Reading analog pin value (temperature) from controller, Num = multiplier and value is *0.161290323*.

After this remapping the temperature value into voltage by using the formula:

$$V = map\ (Temp,\ -40,\ 125,\ 0.1,\ 1.75)$$

Here, V = voltage, Temp = input temperature value, -40 = input minimum temperature value, 125 = input maximum temperature value, 0.1 = output minimum value, 1.75 = output maximum value.

Scenario 3: In this scenario, blinking of LED will takes place when the execution is implemented and running successfully.

- Blink D3 (LED) at 1 Hz to show that the program is executing correctly.
- Turning on D4 (LED) whenever the current exceeds 15A.

Synchronization of Scenario 1, 2 & 3

For the synchronization of scenario 1, 2 & 3 that will work simultaneously we used ***Scheduling*** approach to achieve the desired goal.

Scheduling helps to execute multiple task at a time. For this highest frequency task will execute first because this task have the highest priority and vice versa.

- First count the periods. When the counter is bigger than the threshold it means that the task is active and ready to be executed. Code structure is shown below for the above mentioned statement:

  void scheduler() {

  int i;

  int executed = 0;

  for (i = 0; i < MAX TASKS; i++) {

  schedInfo[i]. n++;

  if (schedInfo[i].n >= schedInfo[i].N) {

  switch(i) {

  case 0:

  task1 ();

  break;

  case 1:

  task2 ();

  break;

  case 2:

  task3 ();

  break;

  ... }

  schedInfo[i].n = 0;

  }

  }

  }

  where, N = number of periods, n = how many times elapsed since the last execution; when it will reach the task let's say 3 and n = 10 than it means that now we should again execute the task 3.

- All the task which are ready are executed within the same orbit (multi task).

3 Results & Discussion

In this section, we will discuss about the results accomplished during the development of the code and real-time implementation on **dsPIC30f4011** board. Figures below, shows the outcomes of the task.

PC to Controller output: From PC microcontroller will receive RPM value using UART. The received RPM is to mapped on the value ranges between 0 to 5V to generate duty cycle.

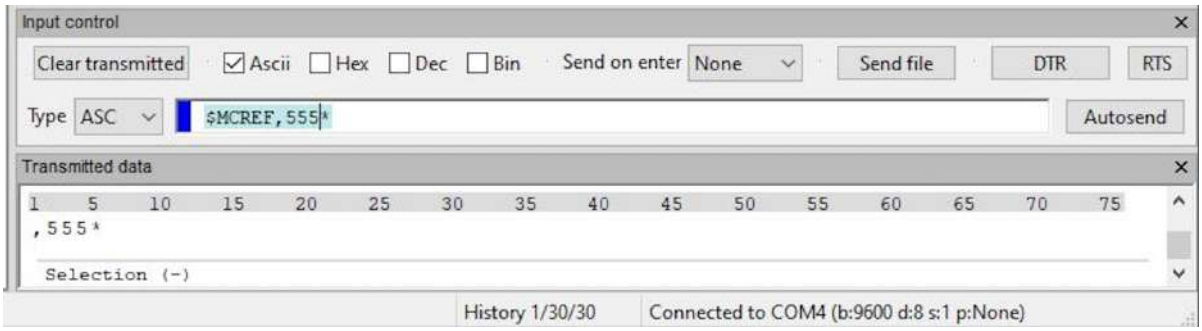

Figure 3: PC to Controller

The PC sends $MCREF,RPM*, where RPM is a value going from 0 to 1000. The references can come as fast as the baudrate allows. As shown in figure 3.

Figure 4: Result1

In figure 4, when we gave the maximum RPM to the motor it shows the the ***Voltage at 5V*** and ***PDC2 at 26614***.

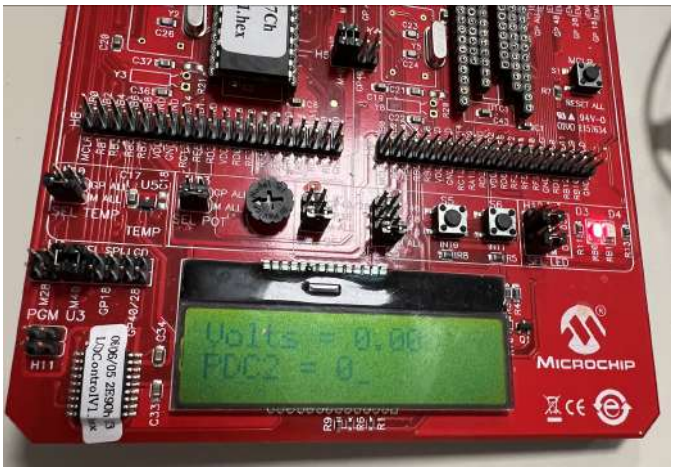

Figure 5: Result2

In figure 5, when we gave the minimum RPM to the motor it shows the the ***Voltage at 0V*** and ***PDC2 at 0***.

Controller to PC output: From microcontroller we receive potentiomemter and temperature values ranges between 0 to 1023. These raw values further converted in volts to generate current and temperature values in degree Celsius respectively.

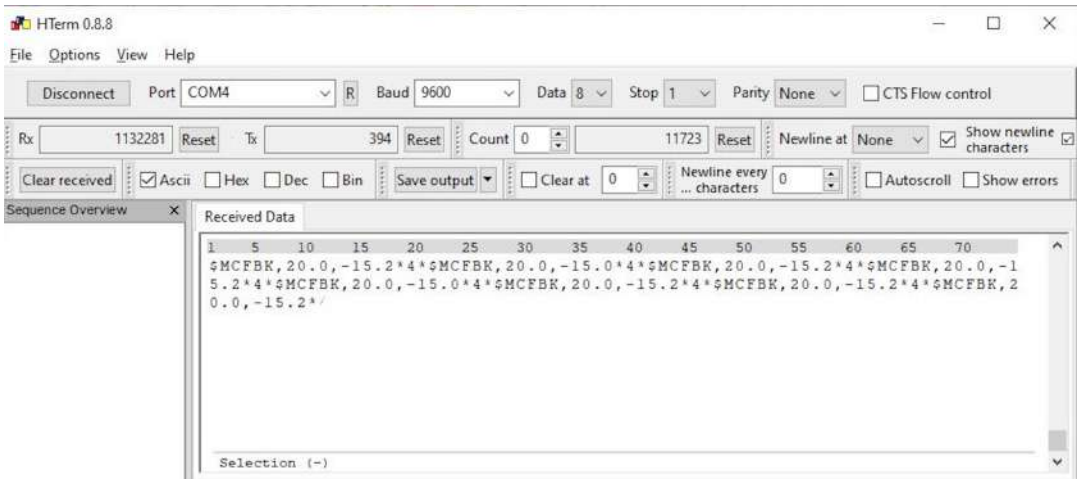

Figure 6: Controller to PC

The micro sends $MCFBK,CURRENT,TEMP*, where CURRENT is the current value in Amperes and TEMP is the temperature value at 1 Hz. As shown in figure 6.

4 Conclusion

The main objective of this assignment is to understand the concept of timers, interrupts, UART, SPI, parser, ADC, PWM, Scheduling and so on. By using the same we achieve the goal of the assign task. We able to receive data from *PC to Controller* and also from *Controller to PC* as shown and discussed under Result & Discussion section.