UNIVERSITY OF TORONTO

INSTITUTE FOR AEROSPACE STUDIES

4925 Dufferin Street, Toronto, Ontario, Canada, M3H 5T6

Robust Robotics

prepared by

Team 67 – Wednesday

Yu Zheng (999911225)
Luyuan Chen (999516082)
Ran(Carrie) Yan (999888126)

prepared for

Prof. M.R. Emami

A technical report submitted for
AER201 – Engineering Design

TA: Rail-Ip

Division of Engineering Science
UNIVERSITY OF TORONTO

1. Acknowledgement

We would like to express our gratitude to all those help us to complete this design project. We are very grateful to our supervisor Prof. M. Reza Emami and TA, Rail-Ip who gave us technical support and encouragement helped all time of the design process as well as their critical advises.

We also appreciate the class of Engineering Science 1T6 and 1T5 for they generously share their experience and resource throughout the entire project to help us overcome many problems. Lastly, we would like to give our sincere gratitude to our friends and family who patiently supported our completion of the project.

2. Abstract

"Robust Robotics" is the robotic prototype that was design and constructed over the entire semester in response to the project proposal, the LED Candlelight Test machine. It is required an autonomous robot that can turn on/off candlelights and detect at maximum 9 candlelights and can efficiently detect the presence of candlelights mounted on the machine, inspect their functionality and record the information gathered. The robot has to count the number of candlelights in testing and identify whether the candlelight is flickering, non-flickering and non-functional. All constraints demonstrated must be satisfied. On top of that, the machine is expected to be reliable, portable and elegant.

This report contains the complete design and construction process of the robot. It also contains the detail description for final prototype of each subsystem including Electromechanical, Circuit and Microcontroller. Further improvement suggestions are given for further interest to strengthen the design. Other related information of robot such as budget, schedule, integration and operating process are included as well.

The machine did not fully perform the design task in the demonstration but was acceptable. It turns on/off all the candlelights expectedly and correctly detect 8 out of 9 functions within 8 (s), which was far below the limited 90 (s). The only failure is due to the calibration mistake in microcontroller code, which can be fixed in 5 minutes. The robot is reliable given that it went through more than 100 times of testing before the final demonstration. Nevertheless, the robot meets all the requirements and constrains, further, some of the bonus features are also implemented.

Table of Content

Intentionally left blank

1. Symbols and abbreviations

Abbreviation	Description
AC	Alternative Current
DC	Direct Current
V	Volt
A	Ampere
mA	Milliamp
GND	Ground
Logic High	5 (V) signal
Logic Low	0 (V) signal
IC	Integrated Circuit
LCD	Liquid Crystal Display
LED	Light Emitting Diode
PIC	Microchip® Peripheral Interface Controller
L Registers	Registers used to store light status
EEPROM	Electrically Erasable Programmable Read Only Memory
I/O	Input/output
UI	User Interface

2. Introduction

Automated quality assurance is one of the most important technic used in all industries. The common requirements for automated function checking machines are accuracy, efficiency, speed and reliability of operation. There are a lot different types available in the market, each one of them are tailored to specialized usage. The client, as a LED candlelight-manufacturing firm, required a machine that can automatically test the functionality of LED candlelights.

"Robust Robotics" is a an automated machine that is designed and constructed over the last 3 months to fulfill the requirement specified in the request for proposal. The prototype is expected to test the 3 functionalities of LED candlelights, namely Pass (candlelights flicker), Flicker Fail (candlelights are constantly on) and LED Fail (candlelights are constantly off). The machine shall be able to test 9 candlelights simultaneously. When the tray is not fully loaded, the machine is responsible for counting the candlelights mounted.

Our machine is a well-designed one, it not only satisfies all the requirement and constraints, but also designed to be elegant, robust and user-friendly. The machine can complete the whole operation within 90 seconds, with test 90.1% accuracy (73/81 tests passed). Moreover, it records the operation time accurate to seconds. It writes up to 4 sets of operational outcome including running time, performance of all candlelights and number of lights to EEPROM, also known as the permanent log.

In the design, workload is divided into 3 subsystems, the electromethanical subsystem, the circuit subsystem and the microcontroller subsystem. Yu Zheng is responsible for the design and construction of the mechanisms and all the structural part of the machine as well as interconnecting the moving parts and sensor modules. Ran Yan is responsible for fabricating all circuits of the machine, including circuit to drive motors, manipulating inputs from the sensor array and the main hub for power and outputs. Luyuan Chen is responsible for the microcontroller coding, designing the logic of the operation and UI.

The first section of the report, the project backgrounds and perspectives are provided. Later in the section, the budget and split of functions in detail are presented in detail. The second section gives specific information on the design of each of the subsystems. Last section contains the integration process, system improvement suggestions, accomplished schedule and conclusion.

3. Market survey

The LED candlelight industry has gone through a boom since they were introduced to the market. The LED lights have lots of advantages over the normal lights, because of its high efficiency in terms of its power consumed (it does not give out much hear as conventional light bulbs do) and it does not radiate harmful material. In the mass-produced industry like LED manufacturing, even the highest final passes yield would generate an enormous amount of units that are not functional. A fast and accurate method of testing each unit is to be implemented. Thus the Automated Test Equipment (ATE) is developed to help the factories accomplish quality assurance quickly and accurately without manpower.

Current existing machines did not match the need stated in the design RFP, for the devices that are being selling are large and expensive. The price listed is typically around $2000 – 50000 CDN on the Alibaba.com. Furthermore, the machines use 380V industrial electricity. The typical machines has high accurate rate and can test over 10k LEDs per hour. In our case, not only this gigantic machine dissatisfied the requirement of the RFP, but also unnecessary to buy one for the purpose specified.

Figure 3.1. Existing industry solution to LED testing

The RFP requires a light capacity test machine, which only take 9 lights each round. Therefore a brand new design shall be proposed to match the requirement of the RFP and the constraints, criteria. The following sections gives the detailed design of the test machine.

4. Objectives and constraints

4.1 Objectives

The machine is required to complete follow objects

- Test 9 candlelights in a single operation
- Identify the functionality of each of the candlelight
- Can operate regardless of the ambient condition
- Easily accessible Emergency Stop button
- LCD should provide information of the operation

4.2 Constrains

- Must fit in 50 ×50 ×50 cm^3 envelope at all operation times
- Weight must be less than 6 kilograms
- Maximum cost: $230 CDN
- Time: entire operation process must be limited in 90 second
- No human interaction after START
- Tray can be separated from the machine and no power supply, actuator or electronic components attached to it
- Safety: no present hazard
- No need for dismantling and installing any part during transporting and loading candlelights

Acceptance criteria in decision-making

AHP Analysis

5. Mechanisms

5.1 Switch gear

As one of the major components of machine, the machine must be able to handle the operation of candlelight's switch. This requires the combination of mechanical and electrical to determine when and how to turn off or on the switch. Currently, some machines have been created to perform similar function. One of the designs will be presented and gained insight of its mechanism below.

Bertho's pointless switch machine

The most important design of this switch machine is the method for controlling the machine to either turn on or turn off the switch. It present the similar function that is needed for our Operating switch gear, because the light must be turned on before the testing and turned off after all the operation.

This machine contains the motor as drive gear to provide the power and it is activated by red pushed bottom. The rotate gear has a small bar used to flip the switch from one side to the other side. When the switch is flipped to another side, it changes the rotate direction of small bar on the rotate gear, and every time when the switch is flipped, the rotate direction will change as well.

Bertho's pointless switch machine is highly effective for the change of rotate direction and switch flipped motion. However, in this machine, the switch is connect to circuit as one part of machine to change the rotate direction, and it is distinctly different from the light switch, which is separated from the switch flipping mechanism. Moreover, Bertho's pointless switch machine uses the complicated circuit without microcontroller to achieve the change of rotate direction. Additionally, the switch of candlelight is small, which means the interaction surface is too small to ensure the bar can be use to flip from one side to the other side. Therefore, this mechanism must be modify including the using microcontroller to simplify the circuit and change rotate direction, the dimension and shape of bar, operation precision.

Figure 5.1.1. Example of the Bertho's pointless switch

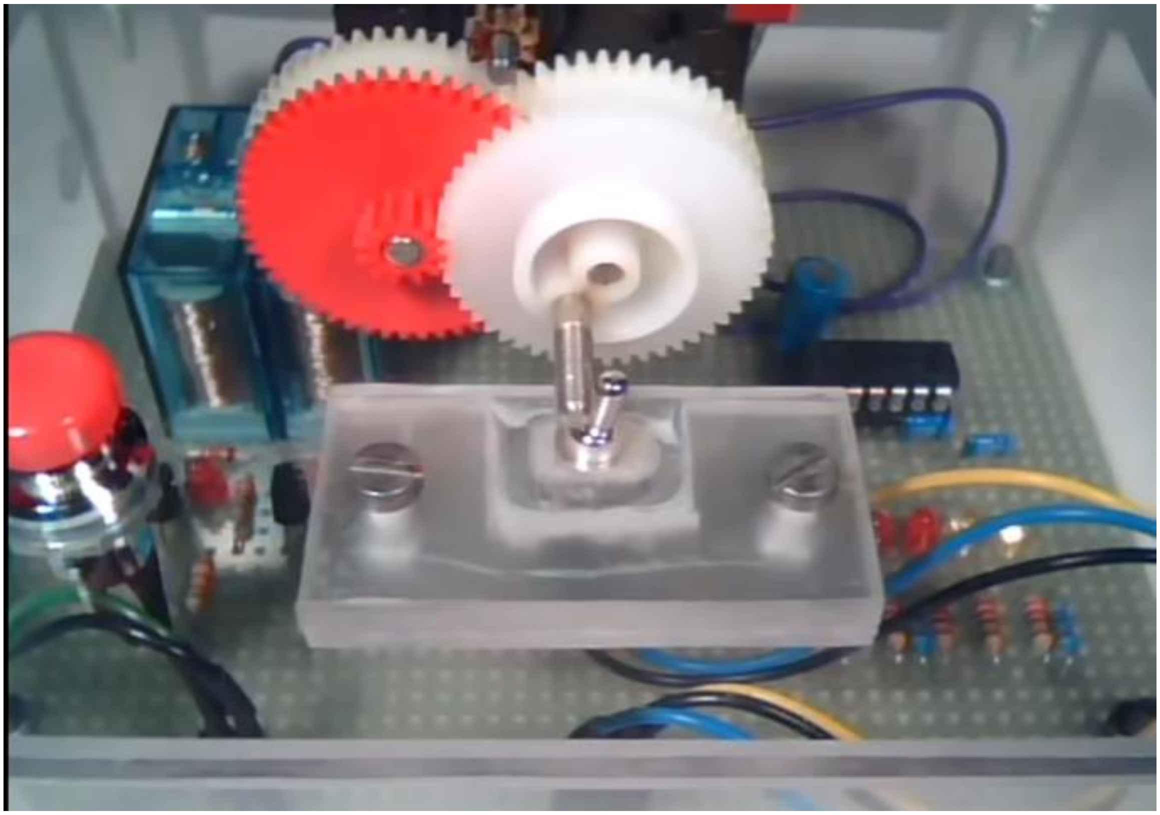


Figure 5.1.2. Example of the Bertho's pointless switch

5.2 Relay

In researching the method to turn on/off the switch, the relay perform the circuit-controlled way to achieve the on-off switch of another circuit. As show in the figure **, the entire system contains two circuit, the right circuit control the switch of left circuit. Once the right circuit is connected, the current go through the spule creating a magnetic force to attract the metal plate above the spule (the iron bar inside the spule is used to reinforce the magnetic force). However, this relay can only produce attractive force, and it is not suitable for the switch of candlelight, which need equal force to turn on/off the switch. But it still provides us with better understanding of applying electromechanical to control the switch.

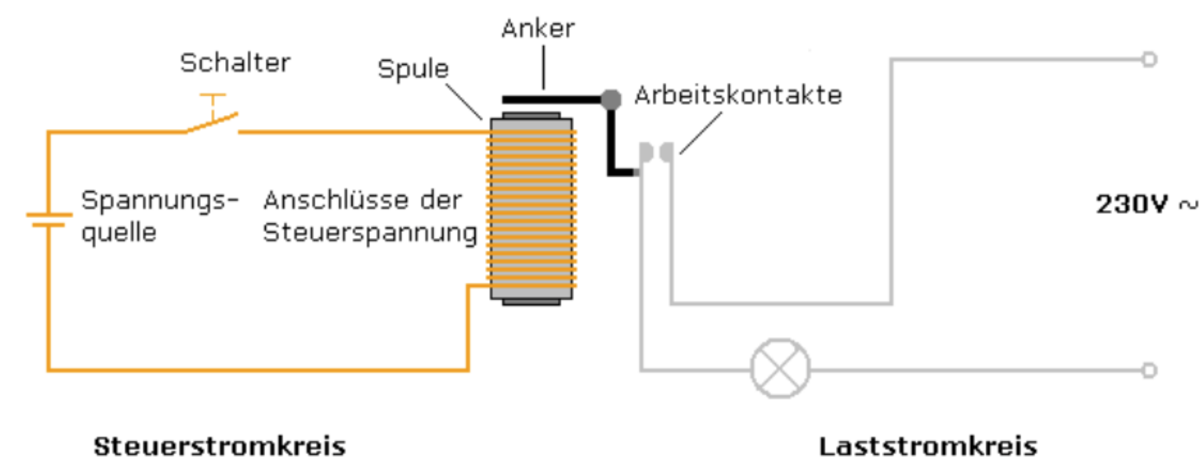


Figure 5.2.1. Example of a relay switch

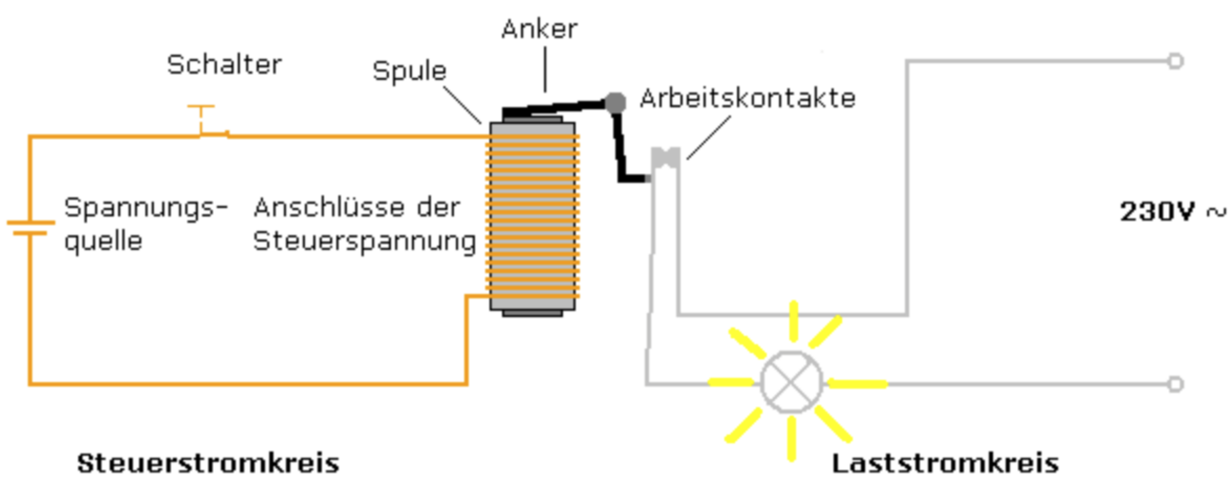


Figure 5.2.2 Example of a relay switch

5.3 Rack and pinion

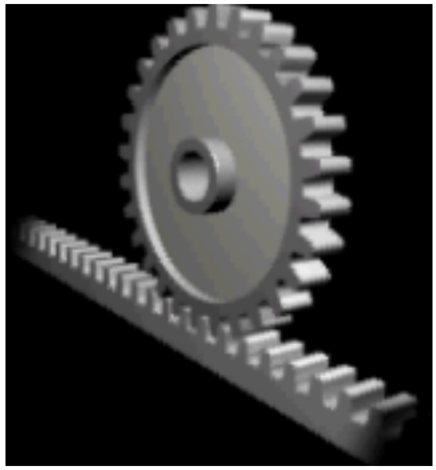


Figure 5.3.1. The rack and pinion mechanism

The rack and pinion is the mechanism that can convert the rotation motion into the linear motion. In other words, this mechanism can change of rotational force such as motor to provide the linear direction force. A rack and pinion gear system is basically the combination of two gears. The pinion is traditional round gear while the rack is the straight gear. The gears are meshed with each other, when one of the gear more, it drives the motion of the other gear. The example of "rack and pinion" gear system can be seen on the rack railway, which is designed to travel up the steep inclines. The mechanism provides the force for train to overcome the large gravity force.

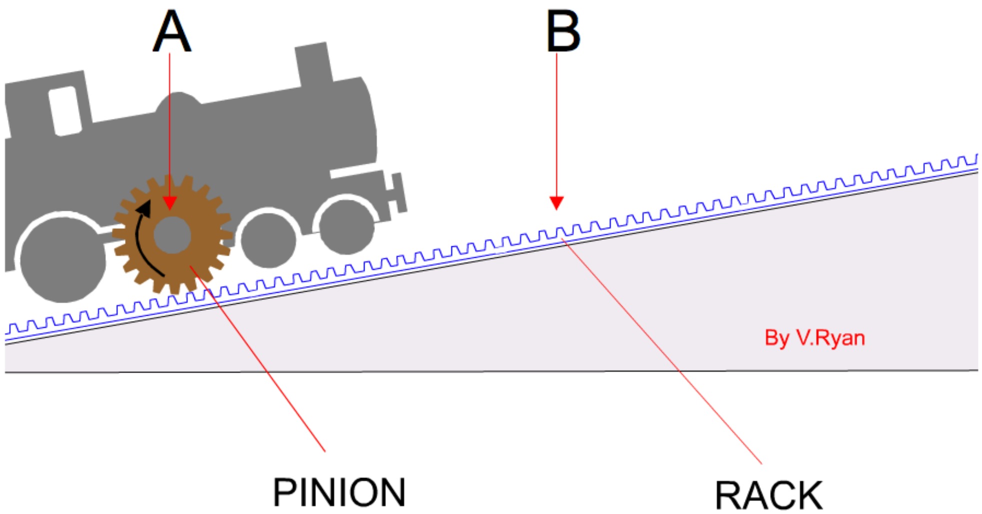


Figure 5.3.2. The application of 'rack and pinion', rack railway

5.4. Theory

The following theories and equations are used for understand the parameters of robot.

Moment of inertia:

Basically the moment of inertia is measure of sluggishness or inertness of a body to the change in its state of rest or rotation. It depends upon mass with respect to axis of rotation. The unit is defined as $kg * m^2$.

Definition for point bodies:

$$I = m\,r^2$$

For the collection of object, the moment of inertia is just the sum or integration over the entire body.

$$I = \sum I = m\,r^2 \text{ Or } I = \int r^2\,dm$$

Torque

The torque is a measurement of how much a force acting on an object causing the rotational motion of object. The object rotates about a fixed axis, called pivot point. The force “F”, and the distance from the pivot to the point where the force acts in called the arm moment arm, label as “r”.

The torque is defined as:

$$\tau = F \times r$$

Where both F and r are the vector, torque is the cross product of these two vector.

Electrical relation is the circuit

Two main formulas is used in the calculation in the circuit part:

$$R = \frac{V}{I}$$

$$P = V\ I$$

where R, V, I and P donated as the resistance, voltage, current and power respectively.

6. Budget

The pricing of all parts are provided in the chart below

Circuit and Microcontroller

Item	Location	Supplier	Number	Unit Price ($)	Total Price ($)
Transistor TIP142N	H-Bridge	Project Kit	2	2	4
Transistor TIP147N	H-Bridge	Project Kit	2	2	4
Diode 1N4001	H-Bridge	Project Kit	4	0.2	1
Capacitor 104	H-Bridge	Project Kit	1	0.3	0
Photodiode BPW34	Light Sensor	Creatron Inc.	9	1.5	14
Operational Amplifier LM358	Light Sensor	Project Kit Creatron Inc.	5	1	5
Socket 2×4 Pins	Light Sensor	Project Kit Creatron Inc.	5	0.30/2	1
Microswitch	Presence Sensor		9	0.3	3
NAND gate 74HC00N	Presence Sensor	Project Kit Creatron Inc.	5	1.2	6
Socket 2×7 Pins	Presence Sensor		5	0.15	1
1×9 Jumper Wires(Male-Male)	Presence Sensor	Creatron Inc.	1	5.3	5
Resistors(1k,5.6k,10k,47k,100k)	All Circuits	Creatron Inc.	40	0.1/10	0
Connecting Wires	All Circuits	Creatron Inc.	20	0.13	3
White Soldering Board (large)	Light Sensor	Creatron Inc.	1	3.5	4
White Soldering Board (small)	H bridge, Presence Sensor, Emergency Stop	Creatron Inc.	3	2.5	8
Green Soldering Board	Light sensor, Signal Selection	Creatron Inc.	2	1.5	3
Emergency Stop Button	Emergency Stop	Creatron Inc.	1	1.2	1
Socket 2×20 Pins	Emergency Stop	Creatron Inc.	2	0.15	0
Multiplexer CD4067BE	Signal Selection	Creatron Inc.	2	1.5	3
Power Supply		Canadian Computer	1	15.5	16
1×9 Jumper Wires(Female-Male)	PIC	Creatron Inc.	3	4.8	14
Black Tape	All Citcuit	Creatron Inc.	1	1.25	1

Devbugger		Project Kit	1	75	75
Total for circuit & microcontroller part $ 170.75					

Electromechanical

Material	Place in machine	
Wood	Switch mechanism, base, tray, motor support	7
Aluminum	Skeleton of the frame, switch mechanism	10
Acrylic	Wall of frame	10
Plastic	Wall of frame	3
Mental plate	Switch mechanism	3
Screw and nut & nail	Frame, switch mechanism	4
DC motor & gear and rack	Switch mechanism	15
Total for electromechanical part: $ 52		

Total Budget for all parts: $ 222.75

7. Division of problem

The construction of entire robot was split into three main composts including electromechanical system, the circuit system and the microcontroller system. All the members should not only be responsible to the design of their own parts but also should help the other members with the debugging works. Before the integration, all the members focus on their work base of the direction of main design of robots. During the integration, the circuit members and programming member helped each other on the calibration of robot while the electromechanical member work with circuit member on design the placement of various circuits and sensor and the DevBugger. All three members discussed the change of idea, improvement problems.

7.1 Electromechanical Subsystem

Task	Member(s)
Before the integration	
General design of robot	All members
Detail design of mechanism	Electromechanical member
Calculation of torque and dimension	Electromechanical member
Selection of motor and construction required material	Electromechanical member
Fabrication of robot	Electromechanical member
During the integration	
Change of design	All members
Calibration of mechanism	Electromechanical member
Placement of sensor and circuit	Electromechanical and circuit members

Table 7.1. Responsibility on electromechanic subsystem

7.2 Circuit subsystem

Task	Member(s)
General design of circuit	Circuit member
Circuit schematics (motor & sensor)	Circuit member
Calculation of resistance, current and voltage	Circuit member
Acquire sensor, power supply and other electric elements	Circuit member
Soldering circuit	Circuit member
During the integration	
Circuit testing	Circuit and programming members
Debugging circuit	Circuit and programming members

Table 7.2. Responsibility on Circuit subsystem

7.3 Microcontroller subsystem

Task	Member(s)
General design of code	Programming member
Writing the pseudo code	Programming member
Pin assignment and user interface	Programming member
Write the code for the motor and sensor	Programming member
Code on the devbugger	Programming member
During the integration	
Calibration of delay for motor and sensor	Programming member
Code for the bonus feature	Programming member

Table 7.3. Responsibility on microcontroller subsystem

8. Electromechanical subsystem

8.1 Introduction

The function of electromechanical subsystem is to provide the overall frame and mechanisms required in the machine. To be specific, the mechanisms in our robot include switch mechanism and candlelight detection mechanism. The electromechanical member is responsible to design the mechanisms of the robot as well as the arrangement of circuit board and power supply. In this section, the entire machine is decomposed into three main parts base on the technological working process of the machine. The detail of design and potential improvement will also be discussed in following.

The overall view of the robot is showed in the **Figure 8.1.1** and **Figure 8.1.2**.

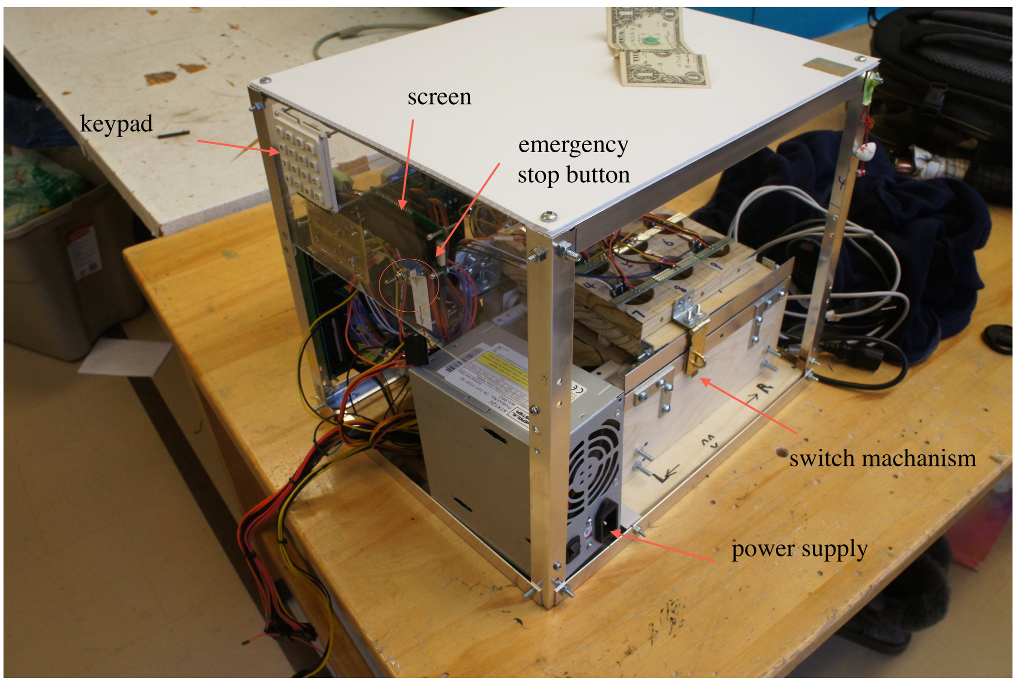


Figure 8.1.1. Overview of the machine

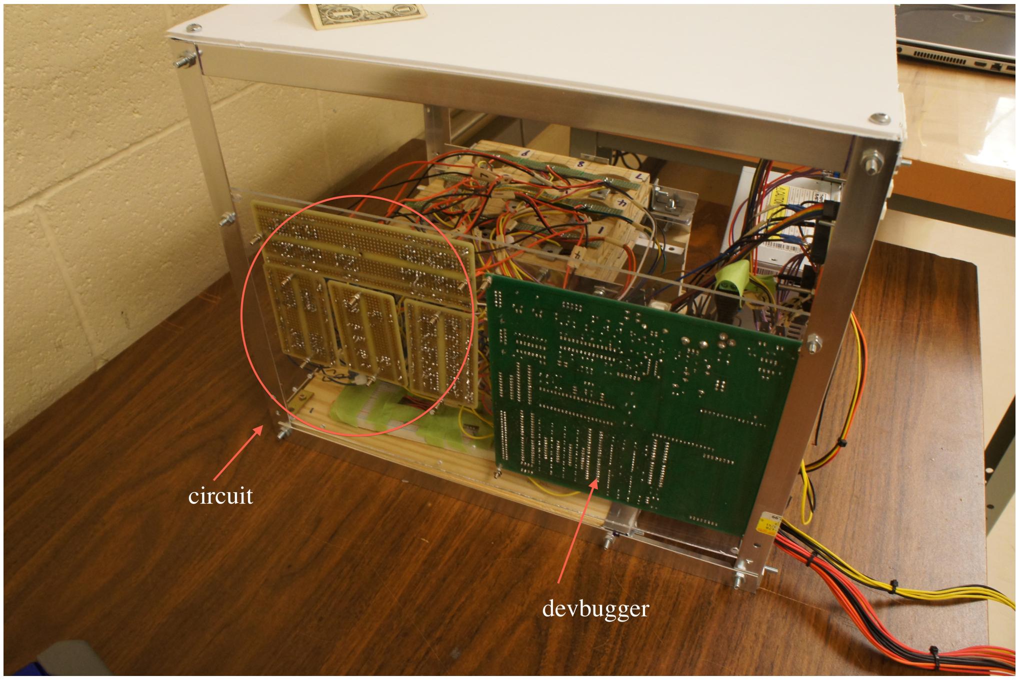


Figure 8.1.2 Back of the machine

8.2 Tray

8.2.1 Assessment of the problem

The tray is served as the platform to hold and fix the candlelights in the prescribed position. If the candlelights cannot be fixed, the candlelights tends to deviate from the original position during the operating process, then the counting number of candlelights and the completing test operation are impossible to finish. The design of tray should be easily assessable to the operator to load in the candlelights and supply the tray to the machine. To sum up, the tray should be reliable to fix the candlelights and convenient to operate.

8.2.2 Solution

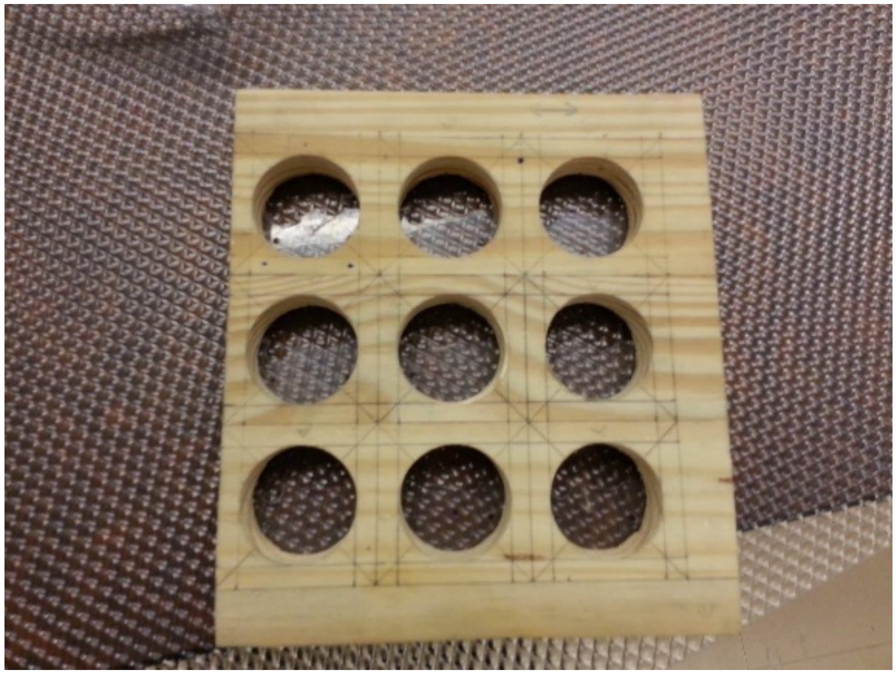


Figure 8.2.1. 3X3 tray

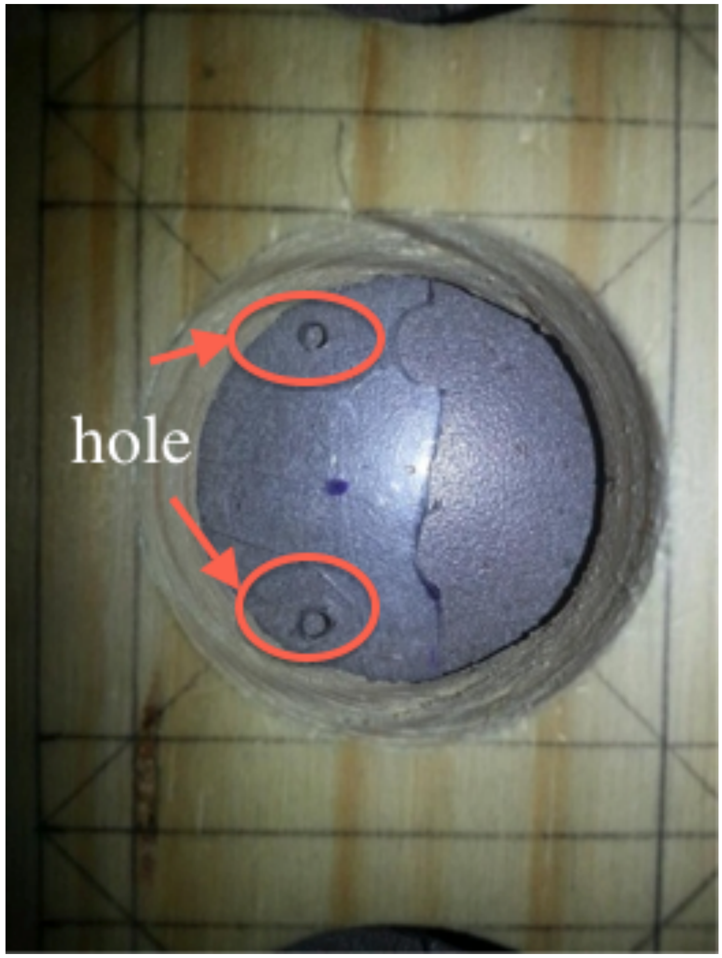


Figure 8.2.2. Grid of the tray

The three by three grid is decided to be used for the tray as showed in the **Figure 8.1.2.** To solve the first problem, which is fixing the candlelight in particular position, the problem is decomposed into three-sub problem. First, to avoid the candlelight move horizontally, the size of each grid is almost the same as the size of the candlelight, so that it can prevent the candlelight move horizontally during operating. Second, to prevent the candlelight rotate in the grid, a special designed base is attached to the tray as found in the **Figure 8.2.2** For each grid, the base contains two holes to allow the stand bar of candlelight to insert inside. The reason why the base is semicircle is another half space is used for the switch mechanism, which implies that the operator who is responded to load the candlelights in the tray should orientate the candlelights to allow the switch of candlelights expose. Third, to keep the candlelight stay in the grid instead of jumping out, the cover in need, the detail design of cover will be discussed in the detection mechanism because it contains the sensors.

8.3 Switch mechanisms

8.3.1 Assessment of the problem

The switch mechanism is the most important part in this robot because it serve the essential function of the robot, turn on/off the candlelights. Without the switch mechanism, the other parts of robot become meaningless. The design of switch mechanism should be reliable and simple. Also, the time spent on turn/off the candlelight should be as less as possible to satisfy constrain that the total operation time must less than 90s. Therefore, the design problem of switch mechanism is identified as following:

- To keep the switch mechanism as simple as possible in order to reduce the mistakes, all the candlelights should be turned on/off at once.
- The friction force tends to cause the jam during turning on/off the candlelights
- The selection of mechanism to drive the force with two directions.

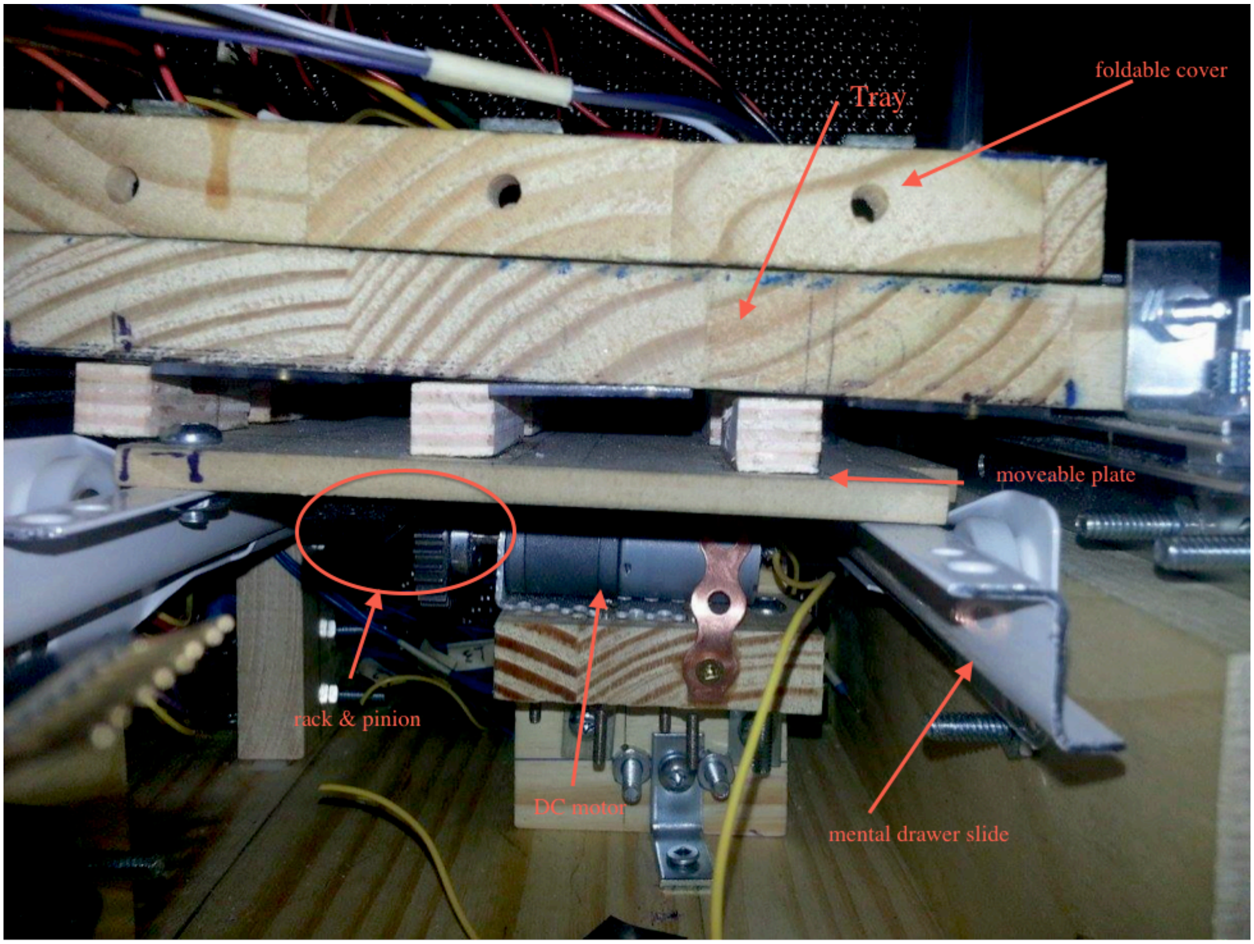


Figure 8.3.1. Switch mechanism

8.3.2 Solution

The entire switch mechanism is showed in the **Figure 8.3.1.** The follow sub-sections will describe the design to solve each problem.

8.3.2.1 Solution to A – moveable base plate

As showed in the **Figure 8.3.1**, there are nine pairs of small blocks sticking on the plate, which is moveable; each pair of wood blocks is in the position exactly under the switch of candlelight. The separation of each pairs of blocks is same to make is possible to turn on/off all the candlelights together. When the tray is supplied in the machine, the exposed switch of candlelights will fall in the gap between two blocks as shown in the **Figure 8.3.2**.. When the operation begins, the plate is drive to the right, and then the left side block hit the switch of candlelight pushing it to right and turn of the candlelight. After finish the detection operation, the plate move back to turn off all the candlelight in the same way.

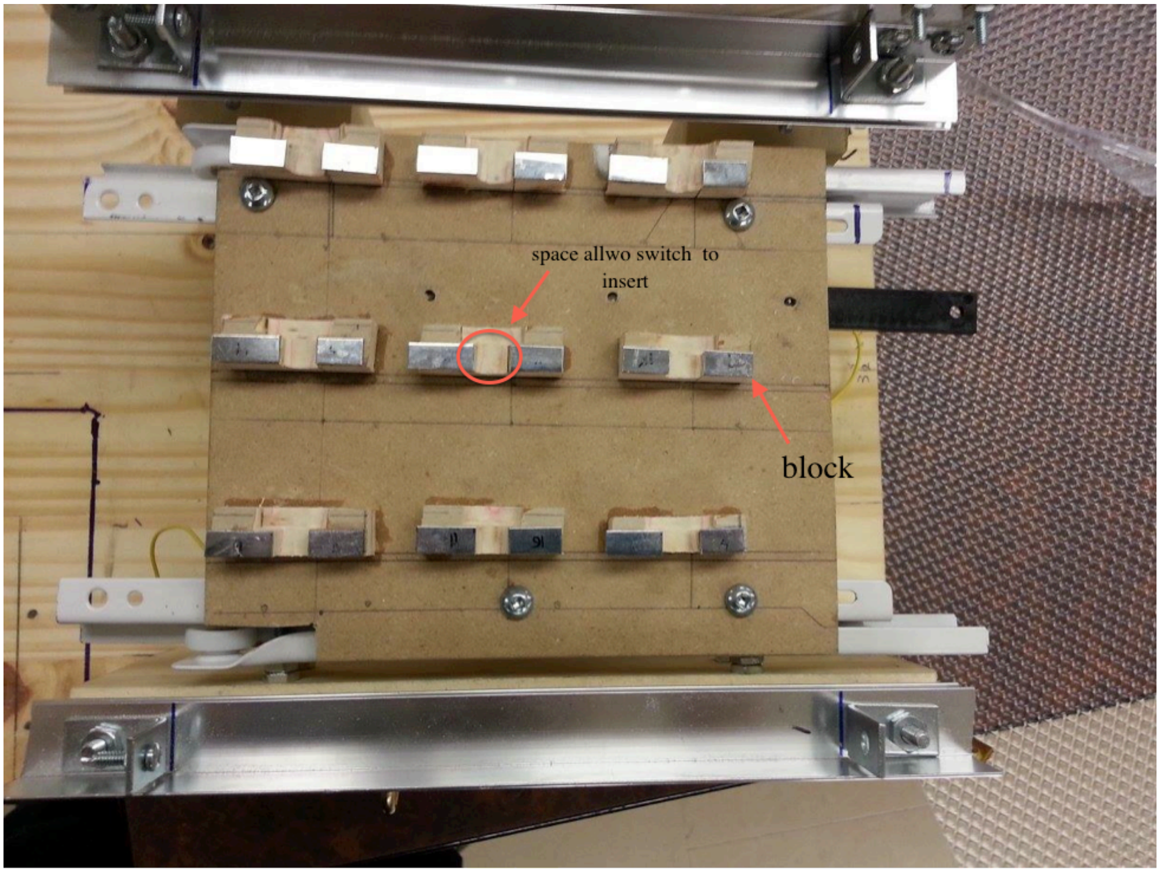


Figure 8.3.1. Moveable base plate

Figure 8.3.2. The tray supplied above the moveable plate

8.3.2.2 Solution to B – mental drawer slide

This idea come from the drawer because the friction to put out the drawer is small no matter how heavy it is. As showed in the **Figure 8.3.2**, two drawer slides is refitted to match the size of our design, between them is connected the moveable base plate. This design can prevent the plate vacillate to left and right when the driving force is provide, and the friction force needed to overcome is reduced sharply.

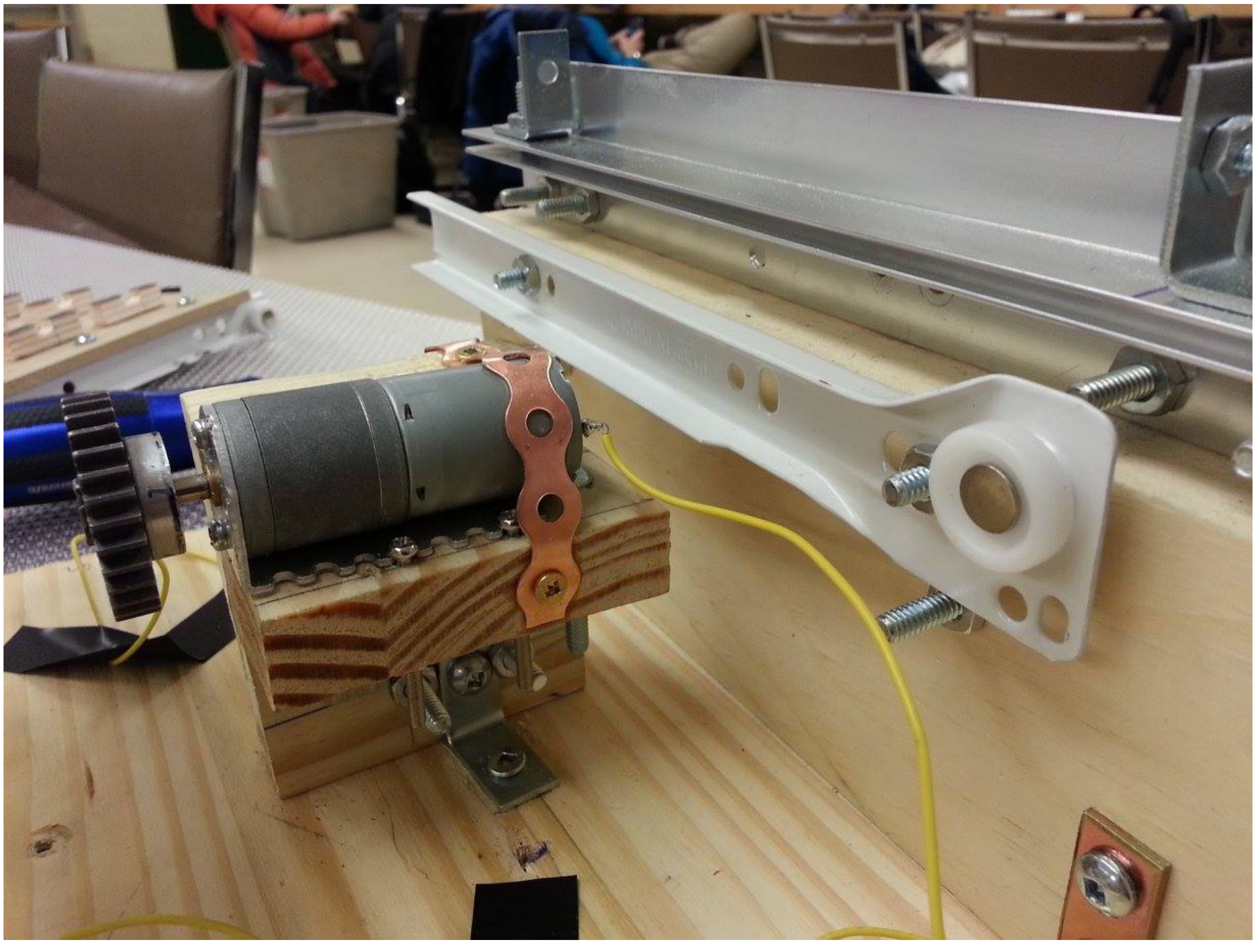


Figure 8.3.3. Mental drawer slide

8.3.2.3 Solution to C – rack and pinion

To provide the horizontal force for both two sides, the linear actuator rack and pinion is employed. The pinion is connected to the motor, when the motor rotates; it causes the pinion drive the rack to move, thereby translating the rotation motion of motor to linear motion of rack. Finally, the rack attaching to the plated and drive the plate moves horizontally for both directions. In the **Figure 8.3.4**, it gives the overall view of this mechanism.

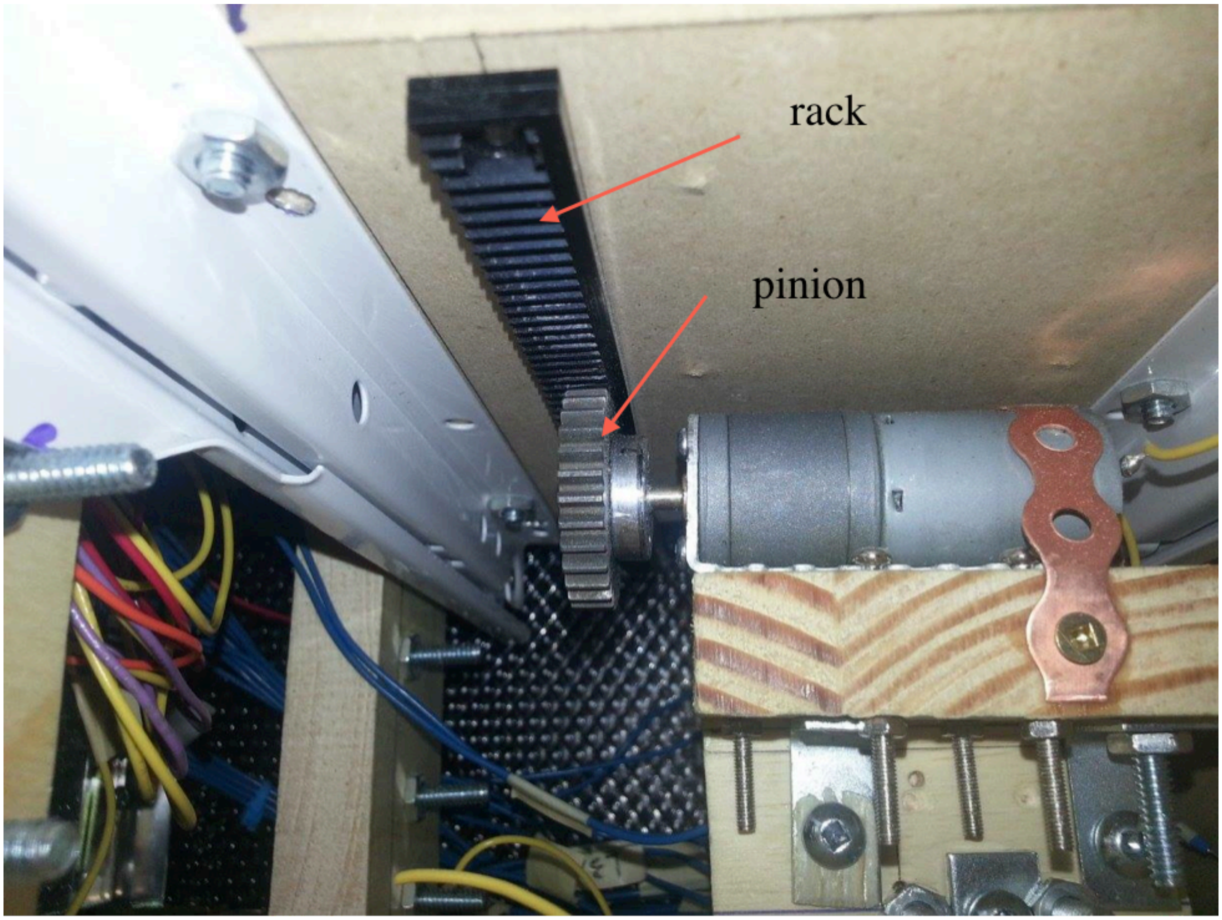


Figure 8.3.4. Rack and pinion

8.4 Detection mechanism

8.4.1 Assessment of the problem

From the RFP, the machine should be able to detect the total number of candlelights and the inspected functionality of each candlelight with three possible states in the test. Again, to keep the mechanism simple, we do not use and moving part here; instead, we use nine sensors to detect nine candlelights respectively. This design focus on the position of sensor and the problem is summarized below:

- Since the light sensor is sensitive, small difference of light intensity may cause the different result of change of sensors. The turn on candlelights may affect each other in the detection process.
- The selection of sensor that can count the total number of candlelights.

- The sensor can only detect the light intensity with no more than 5cm far away from the candlelight. It raises the problem that how to put the tray into the machine without affect the sensor.

8.4.2. Solution

8.4.2.1 Solution to A and C – foldable cover

To separate the candlelight for each grid, we come out the design that using the cover with similar structure of the tray. The cover can fold with hinge attaching to the sidewall. When the cover is close, as showed in the **Figure 8.4.1**, each candlelight can be isolate form other candlelights. Therefore, the sensor will not receive the light from other candlelights. The light sensor is fixed in the middle of grid as shown in the **Figure 8.4.2**. With the folded cover, the tray can put in machine without considering the limitation of space that the distance between light sensor and candlelight must less than 5cm. There is a lock on the cover used to ensure the micro switch is pressed when the cover is closed as well as fixed the tray to prevent the tray moving.

Figure 8.4.1. Foldable cover

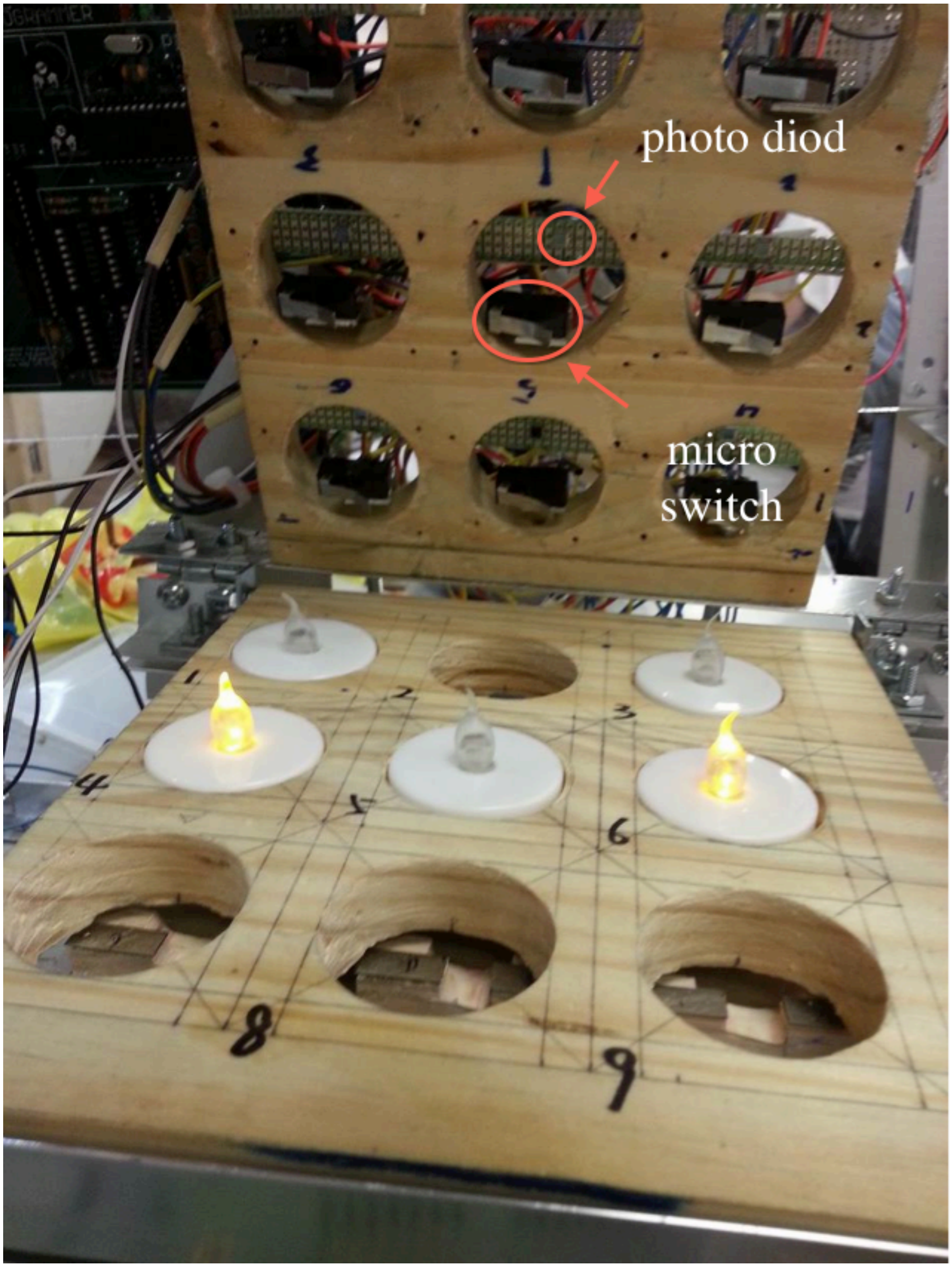


Figure 8.4.2 Sensor in the cover

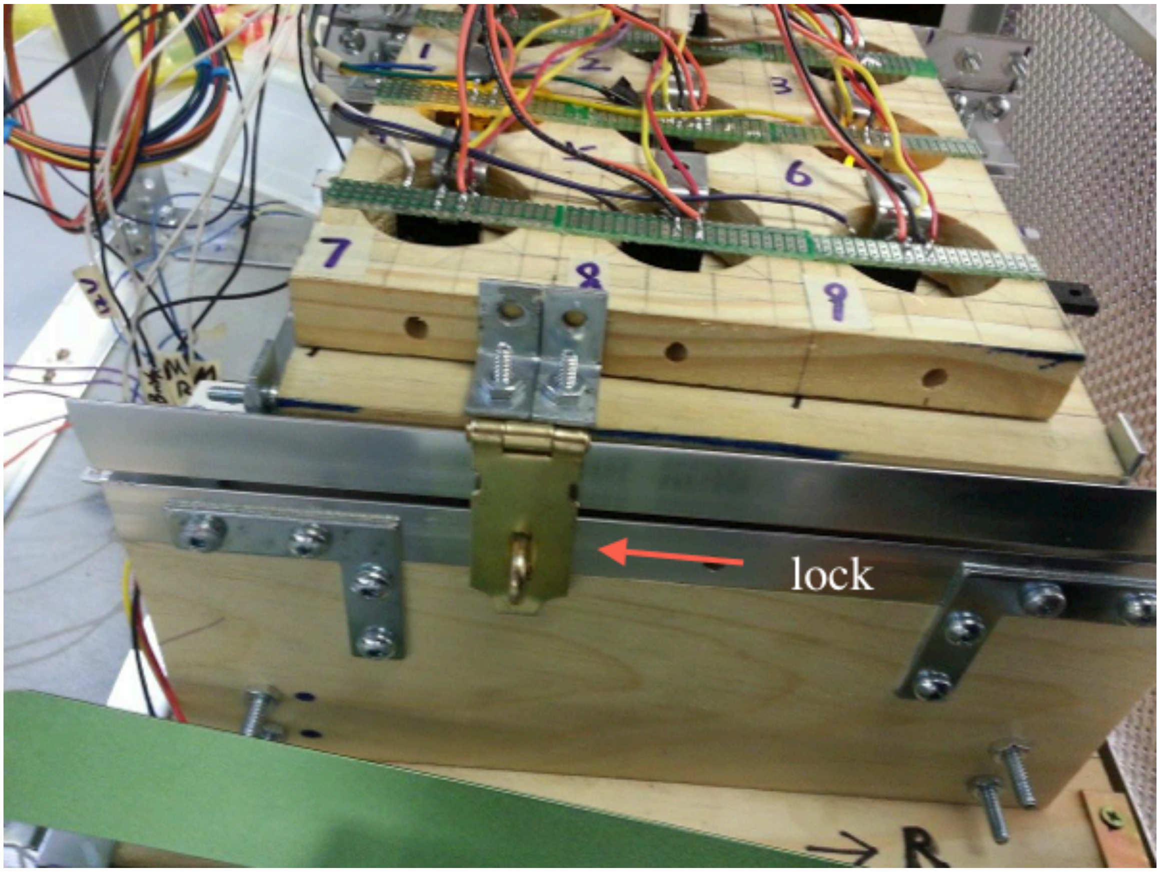


Figure 8.4.3. Lock of the cover

8.4.2.2 Solution to B – Micro switch

The miniature snap-action switch is decided to be used to test the number of candlelight instead of IR sensor is because of its reliability and durability, unlike IR sensor which require high position accuracy. Micro switch only need little physical force to actuate through the use of tipping-point mechanism. **Figure 8.4.4** shows that, the micro switch is attached to the one side of grid of cover. When the cover is fold down, the candlelight in the grid of tray press the actuator of micro switch and create the signal to tell the pic board that there is the candlelight in that position. If the grid of tray is empty, the actuator will not change and no signal created.

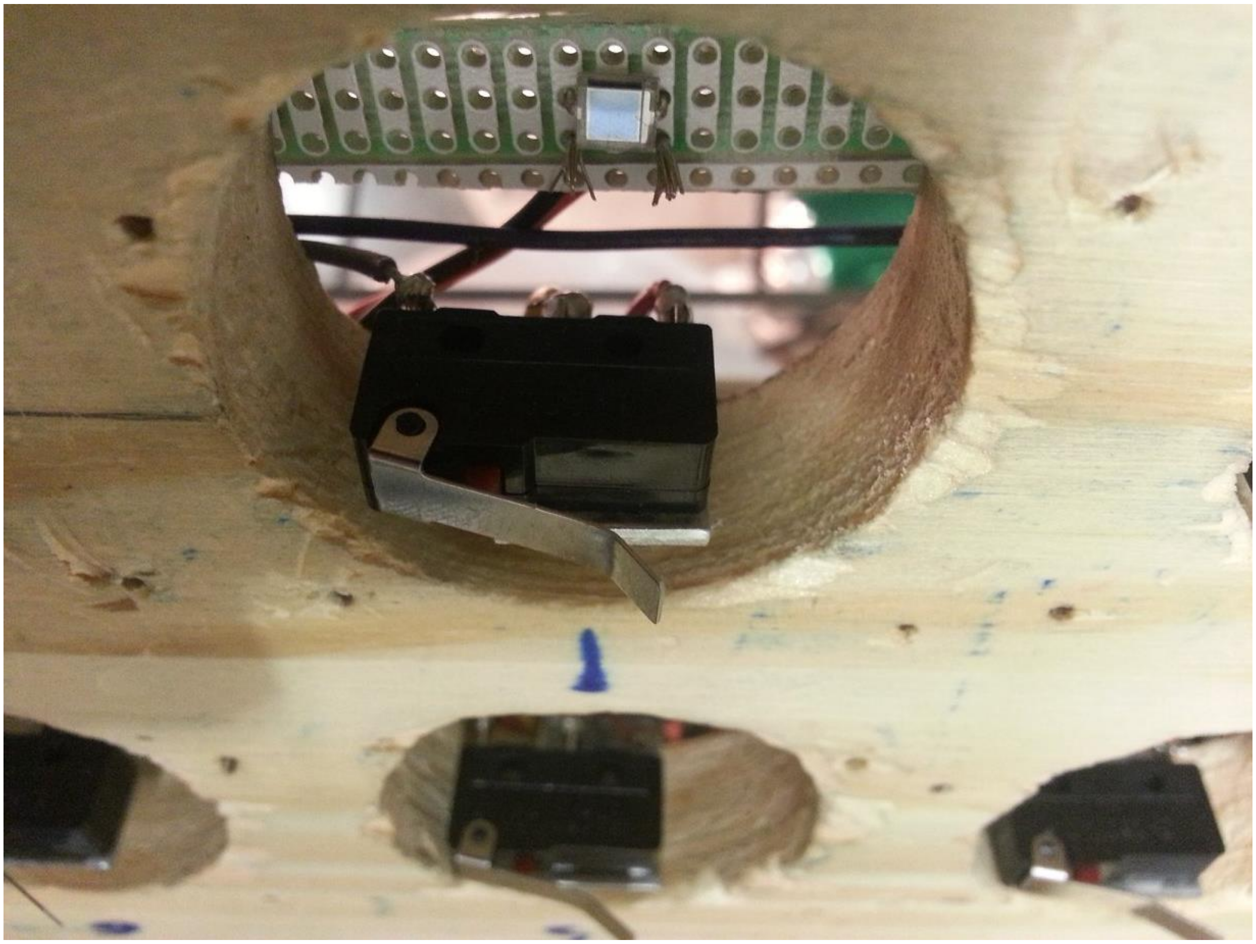


Figure 8.4.4. Micro switch in the cover

8.5 Frame

8.5.1Assessment of the problem

The machine contains not only the mechanisms but also the circuit; the circuit parts contain a large number of electric wires that tend to affect the moving parts. Also, for safety consideration, the frame of machine should contain the wall to prevent the electric wire expose outside causing the potential risk. The material selected for the wall should also be paid more attention, since the machine the design to test the LED candlelight intensity, the light comes from outside tends to affect the sensor. In addition, the material should be as light as possible to satisfy the weight constrain.

8.5.2Solution

Figure 8.1.1 shows that the frame, like a box, encloses all the circuit, PIC board, power supply and the mechanisms in side the frame. All the electric wire is stay above the entire the mechanisms to prevent the wire affect the moving part of machine. The material used as the skeleton of the frame is aluminum, and the wall of the frame is acrylic and plastic. The acrylic is substantial enough that it can be used to support the PIC board and circuit without deformation. The plastic cover is non- transparent so that it can block the light from the environment. The wall is established at four side of machine left one side to allow operator to put in the tray.

All the circuit and the PIC board are fixed on the acrylic board inside the machine. To let the operator more convenient to control the machine, the keyboard and screen are isolated from the Devbugger board and fixed outside the frame as well as the emergency stop bottom as showed in the **Figure 8.5.1**.

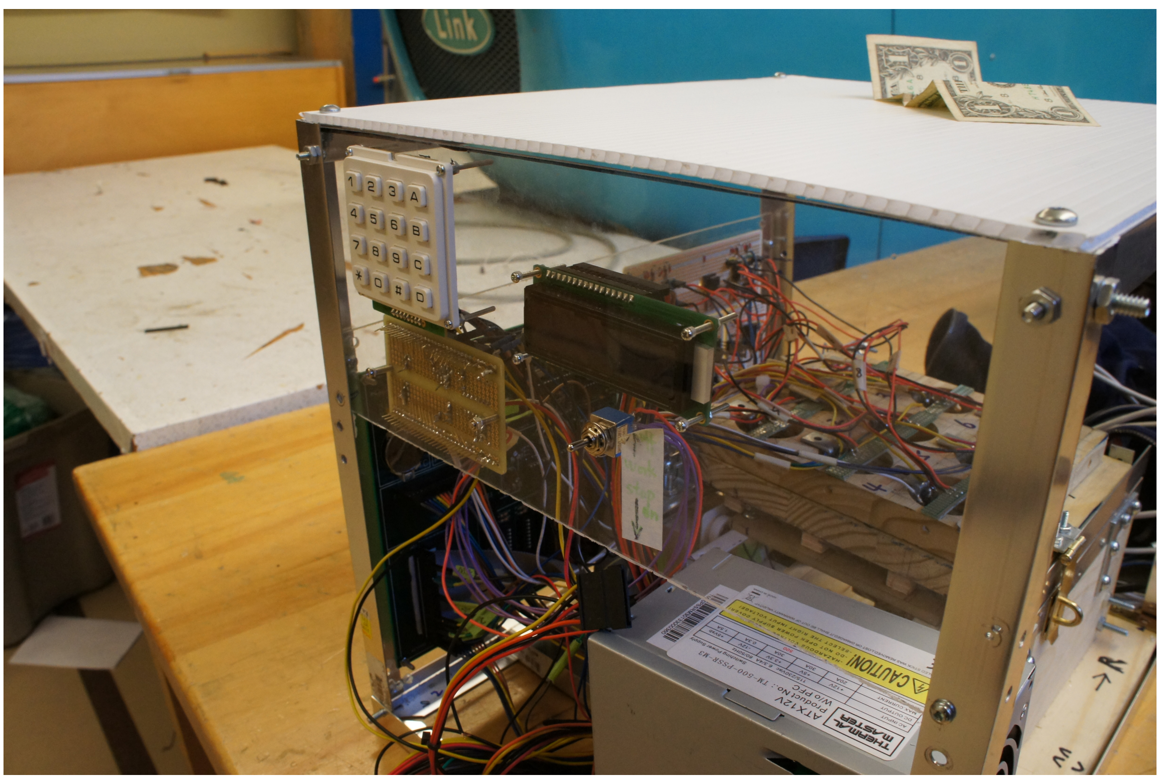


Figure 8.5.1. Keypad, screen and emergency button

8.6 Supporting calculation

8.6.1 Candlelight

- Weight of candlelight: 10 ± 0.1g
- Height without the head of lamp: 1.8cm
- Total height: 3.3cm
- Demeter: 3.8cm
- Distance between random two supporting arms: 2.8cm
- Position of switch: 0.5cm away from one of the supporting arms

8.6.2 Tray

- Size of entire tray: 17cm x 19cm x 1.8cm
- Size of each grid of the tray: 5.3cm x 5.3cm x 1.8cm with cylindrical hole 3.8cm diameter

8.6.3 Moveable base plate

- Distance between two woodblocks on the plate: 1cm

8.6.4 DC motor

- Maximum Force required to turn on candlelights (9 candlelights): 4N * 9 = 36N
- Friction force ≈ 5N
- Total force required = 36N + 5N = 41N = 4.18kg
- Demeter of gear: 1.5cm
- Total required torque = force x length = 4.18kg X 1.5 cm = 6.27 kg*cm
- Selected motor: Zheng DC motor
- Information of motor: 12V, 50rpm, 110oz-in(7.5kg*cm)
- Since 7.5kg > 6.27kg, the DC motor can turn on/off candlelights.

8.6.5 Foldable cover

- Size: 16cm x 18cm x 1.8cm
- Rotatable angle: ≤ 60°

8.7 Suggestion for the improvement of the electromechanical subsystem

8.7.1 Summary of solutions and corresponding actuator

Components of machine	Solution	Actuator
Tray	3X3 squared tray	No use
Switch mechanism	Moveable base plate	No use
	Mental drawer slide	No use
	Rack and pinion	DC motor
Detection mechanism	Foldable cover	No use
	Micro switch	No use
Frame	Box	No use

After testing the robot's functionalities, some improvements can be implemented to improve the robot's reliability and utility rate of space.

8.7.2 Tray

The 3X3 tray limits the cover of tray should also be 3X3 squared which require large space to allow the cover to fold. If the linear tray is applied, less space is need to fold the cover, which means the size of entire machine can be smaller increasing the utility rate of space.

8.7.3 Mental drawer slide

The problem of mental drawer slide is the weight; it contributes most of the weight of switch mechanism. The material used is mental, which has high sturdiness but is not required for this mechanism. The material can change to reduce the weight of machine.

8.7.4 DC motor

DC motor is the only actuator that is connected the rack and pinion, which is used to turn on/off the switch of candlelight. The property of DC motor is that it can only be controlled with its running time instead of the number of rotation. Another property is that, the rotational speed depends on the applied force, the larger force, and the slower rotational speed. These two properties cause the microcontroller member takes a long time to calibrate the time required to turn on/off the candlelight.

In this robot, the time setup for running the motor is constant no matter how many candlelights is used to test. Such that, it cause the problem that if only fewer candlelights is tested, the force required to turn on the switch become smaller, the motor will over rotate and tend to cause the defection of candlelights. The improvement is that, since we test the number of candlelight first before the turning on the switch, the force can change according to how many candlelights inside the tray. This requires more programming from microcontroller member, but it can increase the durability of machine and prevent the defection of candlelights.

9. Microcontroller subsystem

9.1 Overview

The microcontroller acts like the brain of the machine, for it output control command to every electric circuit, and make decision from the information feed to it. The Microchip PIC16F877 microcontroller is chosen to be the only microcontroller used in the project that handles all functions of the candle light test machine. In all, the microcontroller is responsible for interacting with 1 motor controlled by H-bridge circuit (cross cite here), 2 multiplexers (Ti CD4067, Appendix), 9 photo-diodes and 9 microswitches. Additionally, the microcontroller controls several peripheral devices including the LCD display, a 4*4 keyboard, real time clock (RTC), Electric Erasable Programmable Read Only Memory (EEPROM) on the DevBugger V1.2 board.

9.1.1 Problem assessment

The microcontroller should able to accomplish required tasks specified in the request for proposal. There are also optional/ bonus tasks outlined in the RFP.

Required function:

- User interface allows user to interact with the machine in order to fulfill function
- Count the number of lights mounted on the tray
- Identify the functionality of the light mounted
- Display the information of qualified operation

Optional function:

- Real Time Clock (RTC), machine should able to display the time/time elapsed for the operation
- Provide the permanent log function, store at least 4 set of data stored
- PC interfacing, the microcontroller shall transfer the data to PC connected

User interface serves as a means for the machine to communicate with user. User interface shall provide useful information respectively in each phase of the operating cycle. After all the operation done, the user interface should give feedback to the user, displaying comprehensive information of the operation, i.e, the number of lights and the functionality of each light as required by the RFP.

9.2 Solution

9.2.1 Pin assignment of the PIC microcontroller

The PIC16F877 provide 33 I/O ports, most of which are used in the design, the specific pin assignment are shown below, and for further references.

P	#	Buffer	I/O	A/D	Usage
PORTA	0	TTL	N/A	N/A	Light sensor
	1	TTL	N/A	N/A	N/A
	2	TTL	I	A	N/A
	3	TTL	N/A	N/A	N/A
	4	ST	N/A	N/A	N/A
	5	TTL	N/A	N/A	N/A
PORTB	0	TTL/ST	I	D	Emergency switch interrupt
	1	TTL	I	D	Keypad interrupt
	2	TTL	N/A	N/A	N/A
	3	TTL	N/A	N/A	N/A
	4	TTL	I	D	Keypad input
	5	TTL	I	D	Keypad input
	6	TTL/ST	I	D	Keypad input
	7	TTL/ST	I	D	Keypad input
PORTC	0	ST	I	D	Microswitch input (from Mux)
	1	ST	O	D	Multiplexer selection bit D
	2	ST	O	D	Multiplexer selection bit C
	3	ST	I	D	I^2C synchronous, RTC
	4	ST	I	D	I^2C synchronous, RTC
	5	ST	N/A	N/A	N/A
	6	ST	O	D	Multiplexer selection bit B
	7	ST	O	D	Multiplexer selection bit A
PORTD	0	ST/TTL	O	D	Motor control forward
	1	ST/TTL	O	D	Motor control reverse
	2	ST/TTL	O	D	RS (LCD control)
	3	ST/TTL	O	D	E (LCD control)
	4	ST/TTL	O	D	LCD data output
	5	ST/TTL	O	D	LCD data output

	6	ST/TTL	O	D	LCD data output
	7	ST/TTL	O	D	LCD data output
E	0	ST/TTL	N/A	N/A	N/A
	1	ST/TTL	N/A	N/A	N/A
	2	ST/TTL	N/A	N/A	N/A

Table 9.2.1.1. Pin assignment

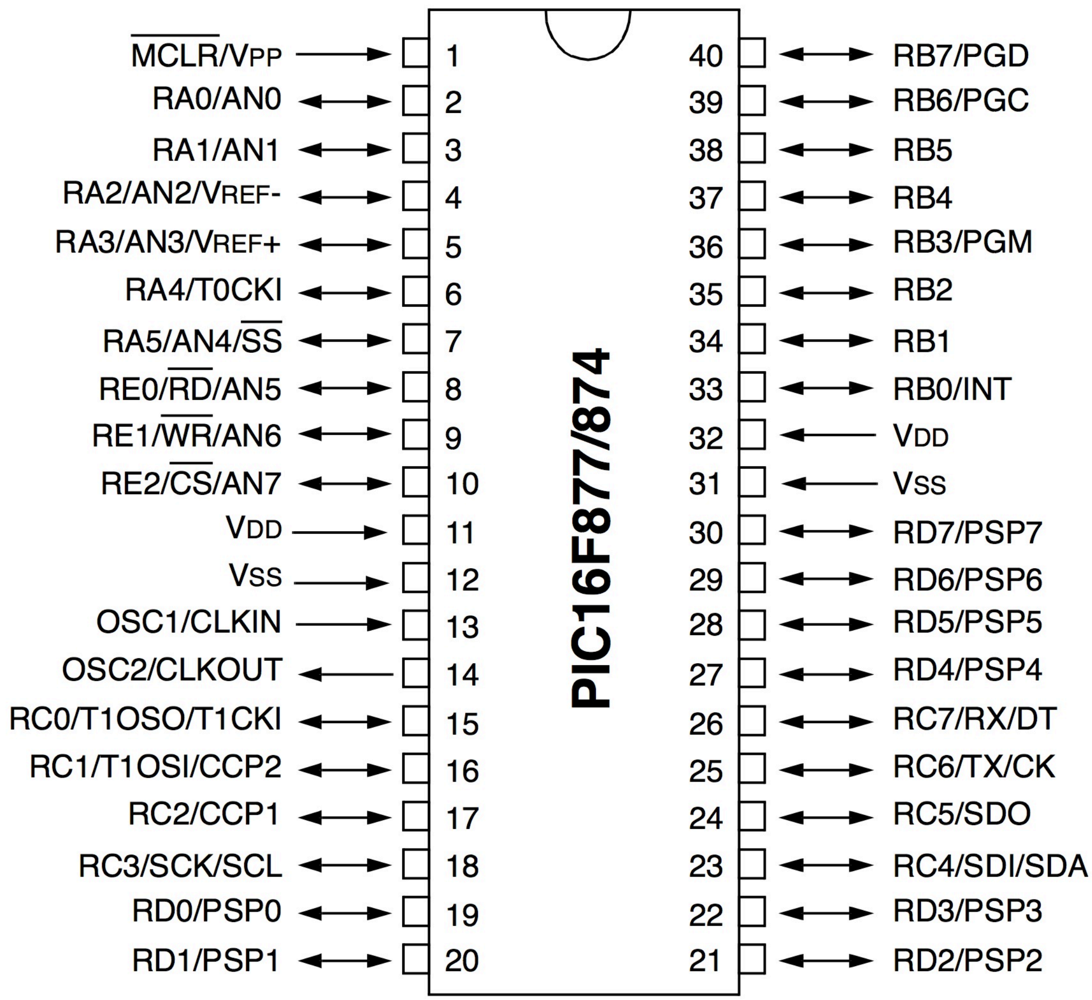


Figure 9.2.1.1. Arrangement of PIC16F877 PDIP packaging chip

9.2.2 User interface

The input from the user is collect by a 4*4 keypad with the MM74C922 encoder, which output a 4-bit signal from the 16-key keypad. The 4-bit digital signal is connected to **PORTB<7:4>** on the microcontroller board. User can follow very simple instruction displayed on the screen to operate the machine. At most of the times, user only have to press either A, B or C. The sample display prompt is given as

Figure 9.2.2.1. Sample screen prompt

The 4-bit binary signal is feed into the microcontroller, and the key detection module will examine the user input. Three keys are used in the project, namely A, B and C. The code only checks the difference between those three keys. After the key pressed by the user is identified, the microcontroller branched to different function modules and sends control to the machine and perform specific task. As shown in figure 1.3.2.1, when A is pressed, the machine enters the working phase (whose detail will be provided in later chapters). B is responsible for the Data logging function required, which gives user the information about the number of lights mounted and the performance/ functionality of each light. The logging module also provide the function to go through up to 4 cycles done in the past by reading the EEPORM. The function of C key is to provide the user an accurate time display; the function is fulfilled by the RTC (real time clock).

1	2	3	A
4	5	6	B
7	8	9	C
*	0	#	D

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

Figure 9.2.2.2 Key selection signal

To output information to the user, a LCD display unit follows HD47780 protocol is used. The PIC microcontroller uses I/O ports to send instructions and data, 4 bits at a time. The complete control mechanism can be found in the **Appendix**.

The LCD display is programmed to show user specific information at different stage of operation. For example, when <A> is pressed to run the machine, the LCD would display the machine is running. After the operation is done, it will display the details about the operation, providing information like the time elapsed in operation and the result. The flow of display on the LCD is shown below.

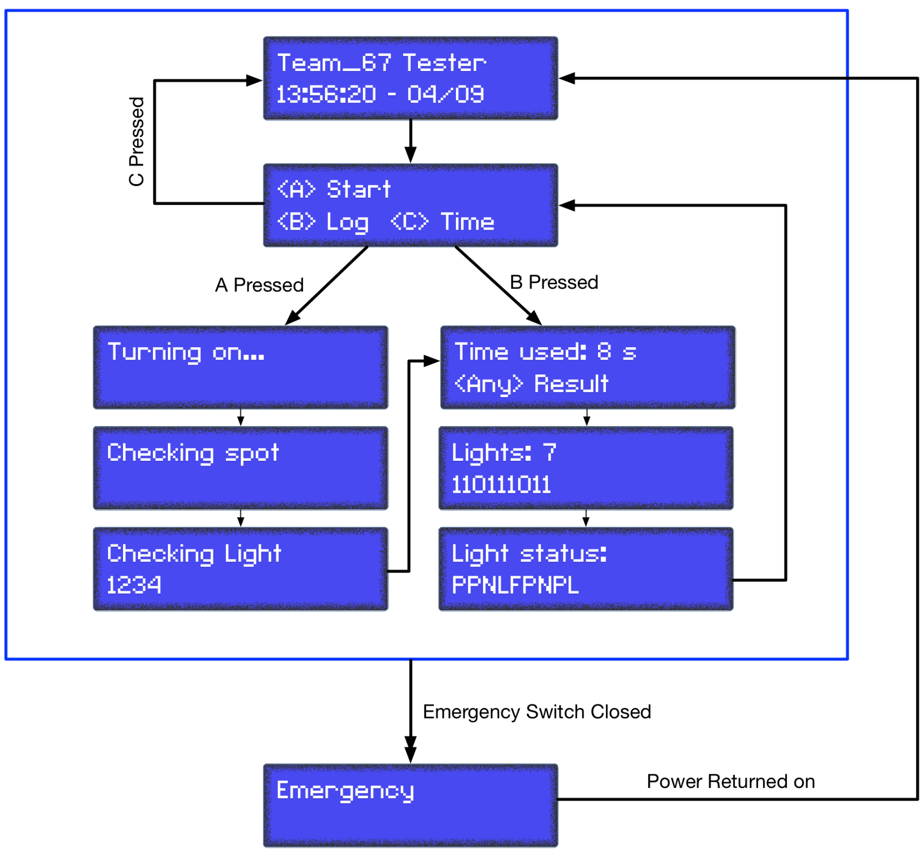


Figure 9.2.2.3. Flow of LCD display

When the machine is initiated or recovered from emergency stop, the machine will display "Team_67 Tester" and the current time and data. After 2s, the screen prompts user to select desired function, whether to start the operation, see the log of recently tested run and to see the time. When the machine enters run mode, the screen will notify user what is running. In specific, it will show "Turning on" when the motor is moving and light is being on; "Checking spot" will be shown when the machine checks the presence of the spots. Then, when the A/D module (see later section for detail) is running, it will show "Checking Light" and the spot number of the spot that is being inspected. After the operation is done, the result page will show the time used in operation. In following screens, the number of lights mounted and the functionality for individual light is presented.

As shown in the flow, the screen will display "Emergency" when the Emergency Button mounted on the machine is pressed. This is done by the interrupt service. Once the emergency button is pressed, the circuit will cut the power of all moving part and send one 5 (V), high signal to **PORTB<0>**, which is the port for external interrupt implemented on PIC16F877 chip. The INTF (RB0/INT external interrupt flag bit) on INTCON is automatically set to 1 and the program will branch to address 0x04 where the interruption code is located. The code snippet for displaying the "Emergency" is located in the interruption code.

9.2.3 Count the number of the lights mounted

There are 9 sets of sensors used in the project, each of which is responsible for one of the spots on the tray (refer to the sensor part for detail). As there are 9 analog input and 9 digital input and the ports are not enough to be assigned, it is decided that the multiplexer (specifically Texas Instrument CD4067 [**Appendix**] 16 to 1 multiplexer/demultiplexer) is used. The multiplexer (referred as "mux" in following text) utilize a 4-bit binary selection signal to choose the channel connected to output. The microswitch circuit is designed to output a 5 (V) (high) signal when the microswitch is closed, the circuit outputs a 0 (V) (low) otherwise. The 9 microswitch circuits is connected to the mux and can be output through the common output on the multiplexer chip. The selection signals are sent from **PORTC<7:6>** and **PORTC<2:1>** while the common output is connected to **PORTC<0>.** The working cycle starts when the selection signal is set, makes the common output connect to the specific microswitch circuit. Then, by reading the status of the RC0 port is the presence of light at that spot determined. In general, when a light is mounted at a spot on the tray, the microswitch on that spot will give a high signal. When the signal is selected by the mux, the RC0 port is pulled high and thus notifies the microcontroller that a light is present in the spot. The circuit is made that has the debouncing function, so the microcontroller is not responsible for the debouncing of the input (with respect to microcontroller). The algorithm of counting is provided below in **Figure 1.3.3**, the flow can be found in **Figure 1.3.5.1** (description of reading all L registers).

```
Check_number
        banksel     PORTA

        clrf        LED_Counter
        btfss       l1, 0   ; skip if the 0 bit of l1 is set, add counter otherwise
        incf        LED_Counter
        btfss       l2, 0   ; skip if the 0 bit of l2 is set, add counter otherwise
        incf        LED_Counter
        btfss       l3, 0   ; skip if the 0 bit of l3 is set, add counter otherwise
        incf        LED_Counter
        btfss       l4, 0   ; skip if the 0 bit of l4 is set, add counter otherwise
        incf        LED_Counter
        btfss       l5, 0   ; skip if the 0 bit of l5 is set, add counter otherwise
        incf        LED_Counter
        btfss       l6, 0   ; skip if the 0 bit of l6 is set, add counter otherwise
        incf        LED_Counter
        btfss       l7, 0   ; skip if the 0 bit of l7 is set, add counter otherwise
        incf        LED_Counter
        btfss       l8, 0   ; skip if the 0 bit of l8 is set, add counter otherwise
        incf        LED_Counter
        btfss       l9, 0   ; skip if the 0 bit of l9 is set, add counter otherwise
        incf        LED_Counter
        return
```

Figure 9.2.3.1. Code for counting LEDs

9.2.4 Identify the functionality of the light mounted

It is requested by the RFP to identify three status of the LED candlelight, flicker failure (the light is constant on), led failure (the light is constant off) and pass (light flickers). The identification cannot be done by digital signal input, so the Analog-to-Digital Converter Module built in the microcontroller is used. As discussed in the previous section, the signal of sensor is feed from the multiplexer. The sensor chosen is photo-diode which is very sensitive to light intensity. To be specific, it gives a large range of voltage output when in different environment. The sensitivity of photodiodes gives makes it easy to utilize A/D module to assess the functionality of the lights.

The A/D converter transforms an analog signal into a 10-bit digital signal. To be specific, the A/D converter works at a specific frequency, which is the sample rate. The 10-bit binary signal is split into two registers, which are ADRESH and ADRESL. The voltage of input signal is often around 2.5 (V) so only the low eight bits stored in the ADRESL is checked. The working scheme of the A/D module in the case of ours in as follows, details about the module is also discussed below.

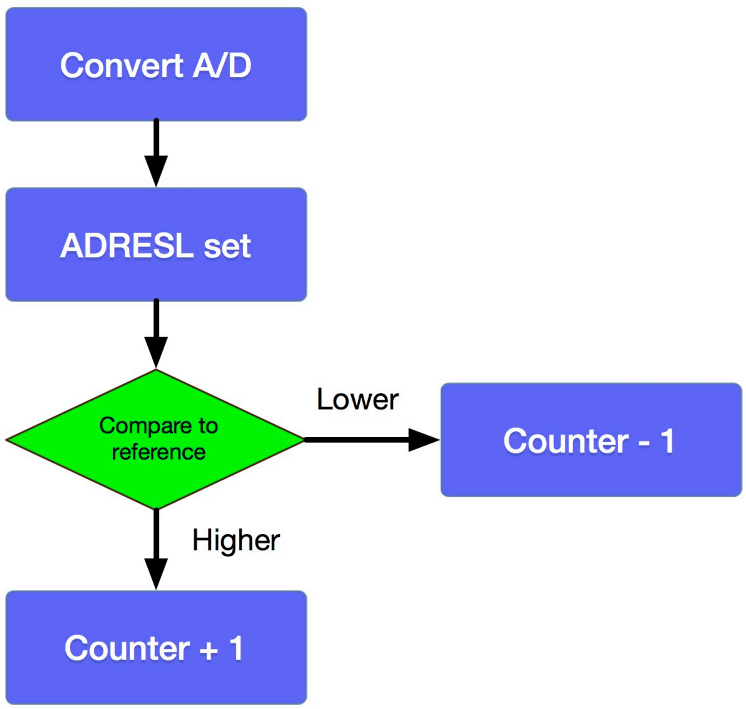


Figure 9.2.4.1. Work flow of the A/D module

The voltage range of output from photodiode placed on a flickering light would be 0.9 to 3.0 (V) in our case. It is possible to select a voltage in between to act as the reference voltage. The reference voltage is carefully selected and calibrated for each spot on the tray to provide most accurate measuring. In one operation cycle (checking one light), 99 data point is taken consecutively with an interval of 0.01 (s) each. Then the voltage of each point is compared to the reference voltage. As shown in the flow chart, there is a register called "Counter", which is set to 100 when initialized. When the light has a status of "flicker failure", or the light is constant on, the voltage input will is always higher than the reference voltage resulting in the counter get accumulated to 199. In contrast, in the case of LED failure, the counter will be 1 after the operation because of the fact that the voltage input will be constantly lower than the reference. Lastly, anything in between would dictate a flickering light (as "PASS"). Thus the status of light can be identified.

To store the information collected from the sensor, 9 separate registers are used. They are named from L1 to L9; each one of them follows a specific format convention:

Pass	Flicker failure	LED failure	No light
0b1000	0b0100	0b0010	0b0001

Table 9.2.4.1. Format convention of storing light status

The one-hot coding would make it easy to check and read. The 9 registers are initialized to be 0b0001 assuming no light is mounted at all spots. After making sure there is no light present at the spot, the code is change to 0b0000 for further changes. The specific flow is shown below.

```
setInitialValueForLightResigters
    movlw       b'0001'         ; LSB: 0001 as initial value: constant off light
    movwf       l1

    btfsc       PORTC,0         ; skip if the port is not set, proceed if the port is set
    bcf         l1, 0           ; clear the 0 digit of the l1 register

;****************************************
; Check functions
;****************************************
regLight_1
    ; check the led register and log the info onto the l* register
    ; the l* register already cleared if the light is present
    btfsc       LED, 1  ; skip if the 1st bit not set, set l* register otherwise
    bsf         l1, 1
    btfsc       LED, 2  ; skip if the 2nd bit not set, set l* register otherwise
    bsf         l1, 2
    btfsc       LED, 3  ; skip if the 2nd bit not set, set l* register otherwise
    bsf         l1, 3
    return
```

Figure 9.2.4.2. Code for manipulating data storage

Also, to minimize the time used in operation, when the spot is empty, the most time consuming A/D conversion phase is skipped.

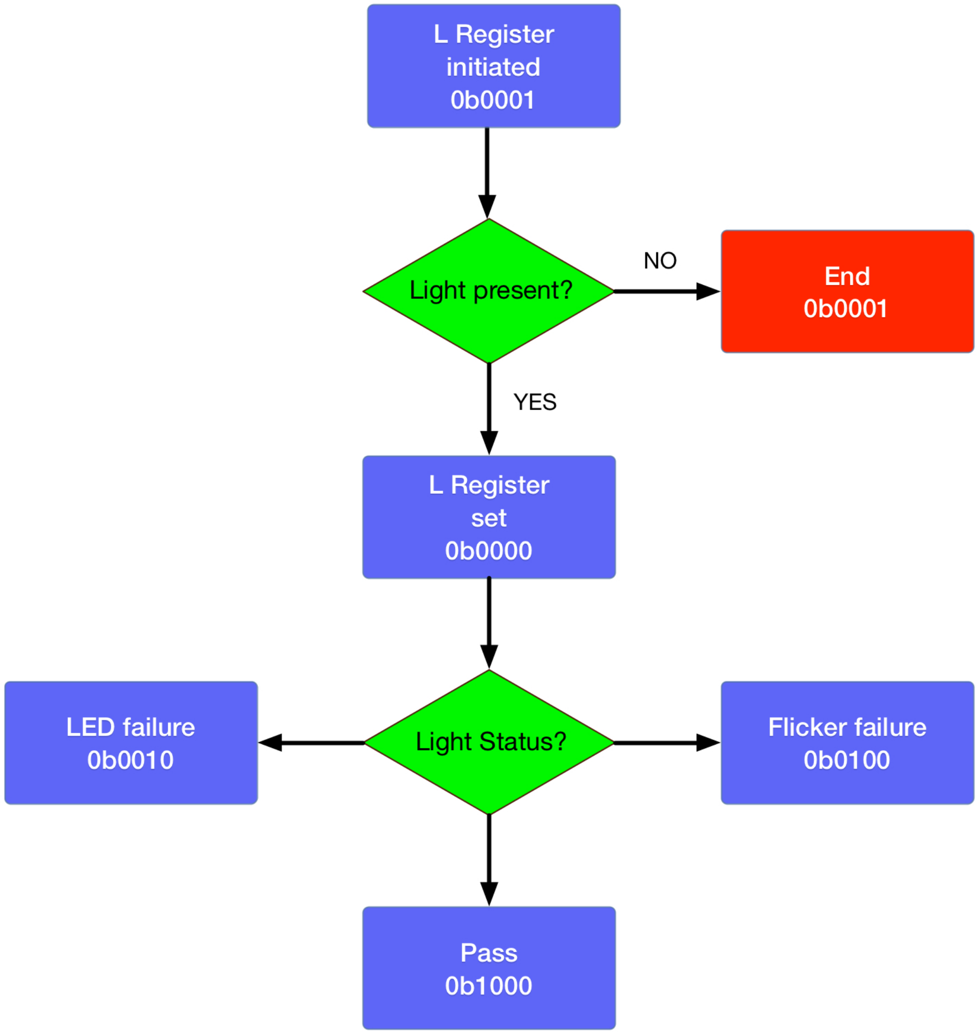


Figure 9.2.4.3. Schematic of storing data in memory

9.2.5 Display the information of qualified operation

At the end of each operation, the LCD display would display the information of: time used in the operation, number of light and status of presence of the operation and the functionality of lights. The result reading is done by checking the bits of L1 to L9 registers (discussed above). Working flow accompany with the code for displaying message is provided below.

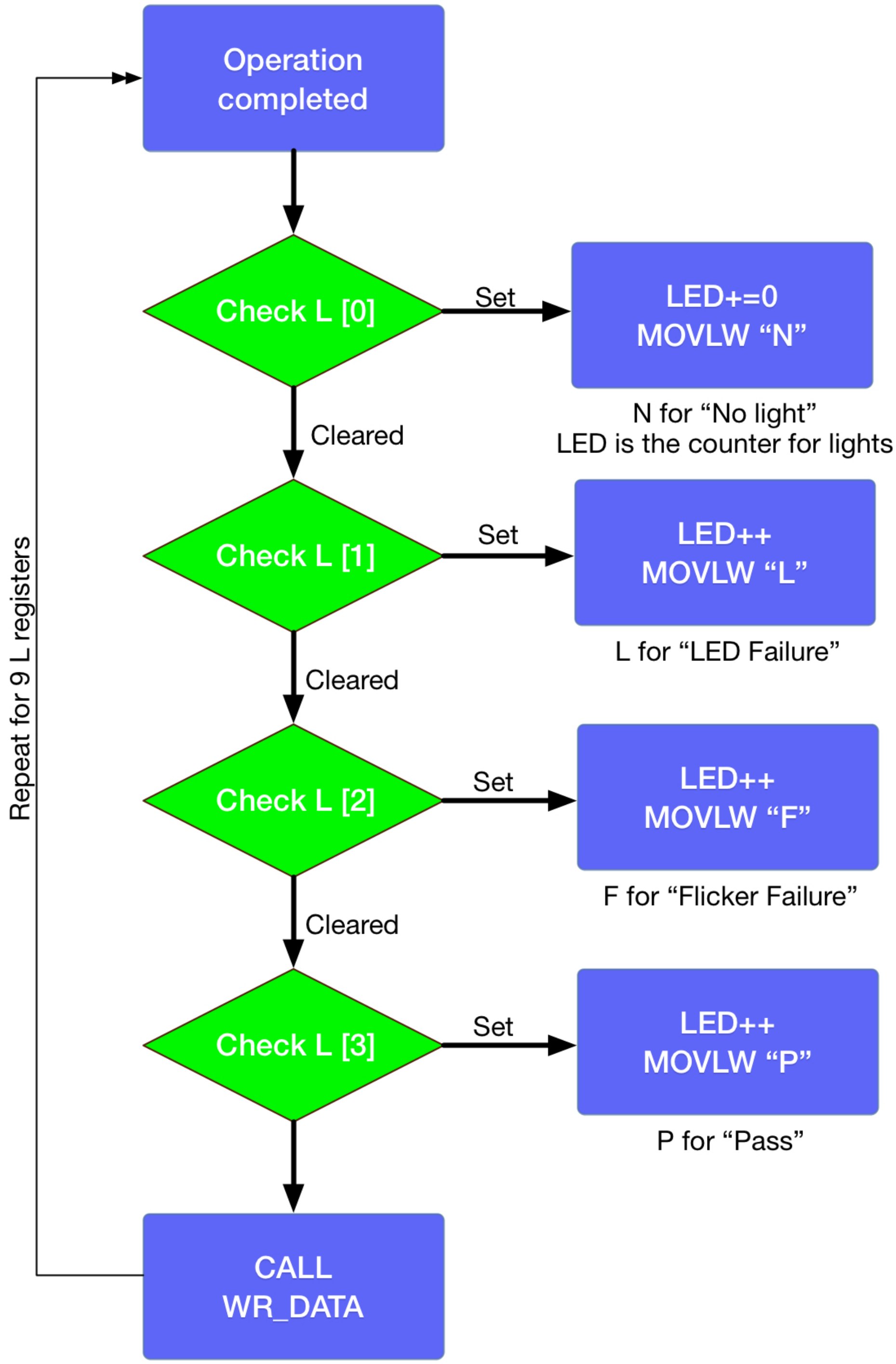


Figure 9.2.5.1. Flow of assessing L register

```
;****************************************
; Display macro
;****************************************
Display macro   Message
        local   loop_
        local   end_
        clrf    Table_Counter
        clrw
loop_   movf    Table_Counter,W
        call    Message
        xorlw   B'00000000' ;check WORK reg to see if 0 is returned
        btfsc   STATUS,Z
        goto    end_
        call    WR_DATA
        incf    Table_Counter,F
        goto    loop_
end_
        endm
```

```
    ;****************************************
    ; Write data to LCD - Input : W , output : -
    ;****************************************
WR_DATA
    bsf     RS
    movwf   dat
    movf    dat,w
    andlw   0xF0
    addlw   4
    movwf   PORTD
    bsf     E               ;
    call    lcdLongDelay    ;__    __
    bcf     E               ;  |__|
    swapf   dat,w
    andlw   0xF0
    addlw   4
    movwf   PORTD
    bsf     E               ;
    call    lcdLongDelay    ;__    __
    bcf     E               ;  |__|
    return
```

Figure 9.2.5.2. Code to write to screen

As can see in the code, the display module uses a table entry for data, write it separately in a loop to display whole messages.

9.2.6 Real Time Clock

A DS1307 real time clock chip with a 3 (V) battery forms the basis of the real time clock function. The microcontroller communicates to the chip by the I^2C communication protocol on I/O ports **PORTC<3:4>**. The specifications of the DS1307 chip are given in the **Appendix**. The real time module provides two functions: provide real time and act as timer.

The real time clock is initialized to the real time at it is first programmed. Then with the additional power supply, it runs regardless of the power of the board. When used as a timer, the time stamp is recorded at the beginning of each operation cycle. After the whole cycle is done, another stamp is recorded. The time elapsed of the operation is the difference between the two stamps. Furthermore, by converting the number into ASCII code, it is able to output the number onto the screen.

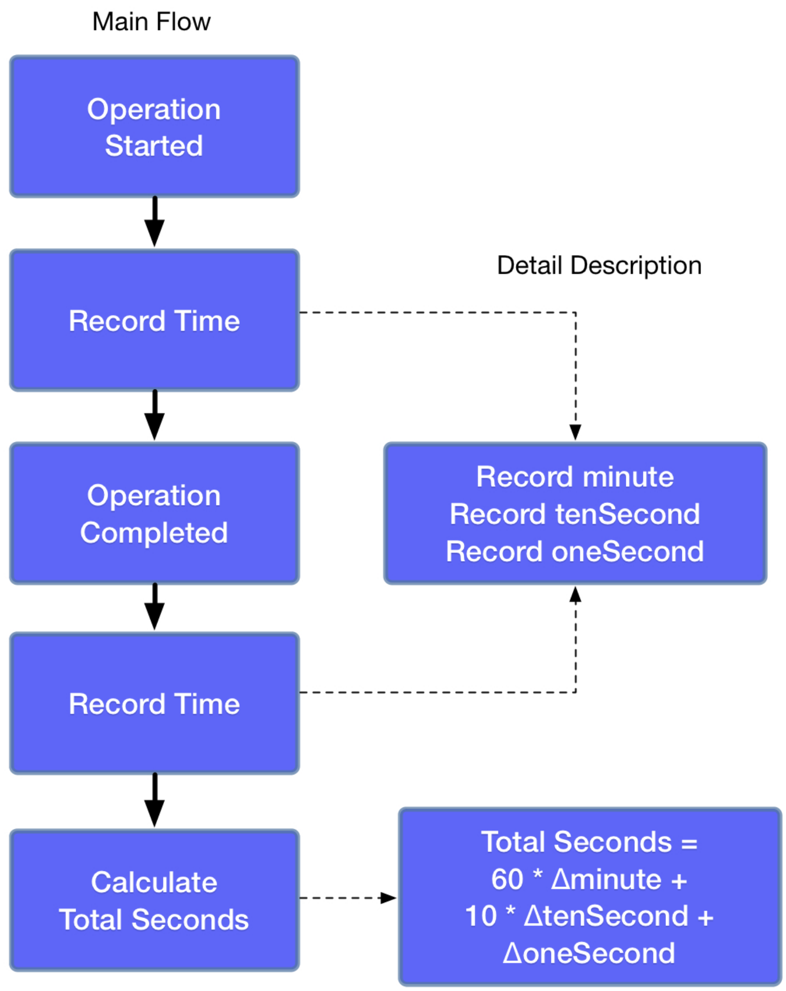


Figure 9.2.6.1. Algorithm of Real Time Clock

```
;****************************************
; Setup RTC with time defined by user
;****************************************
set_rtc_time

        rtc_resetAll    ;reset rtc

        rtc_set 0x00,   B'10000000'

        ;set time
        rtc_set 0x06,   B'00010100'     ; Year
        rtc_set 0x05,   B'00000010'     ; Month
        rtc_set 0x04,   B'00101000'     ; Date
        rtc_set 0x03,   B'00000001'     ; Day
        rtc_set 0x02,   B'00010011'     ; Hours
        rtc_set 0x01,   B'00100101'     ; Minutes
        rtc_set 0x00,   B'00110000'     ; Seconds
        return
```

Figure 9.2.6.2. Time setting of RTC, which is only needed once before battery runs out

```
checkTimeElapsed
    ; first check the minute by subtract them
    clrf    totalSecond     ; init the register
    ;Get minute

    rtc_read     0x01       ;Read Address 0x01 from DS1307---min
    movfw        0x78        ; minute right now
    movwf       Temp
    movfw       minute      ; minute read when starting in W
    subwf       Temp, w     ; place the result in w
    call        multiplyByTen
    call        multiplyBySix        ; time the minute by 60 -> second in W
    movwf       totalSecond

    ; calculate second
    rtc_read     0x00        ; read the seconds
    movfw       0x77        ; 10digit to w
    movwf       Temp
    movfw       tenSecond
    subwf       Temp, w     ; place the result in w
    call        multiplyByTen
    addwf       totalSecond, f

    movfw       0x78        ; 10digit to w
    movwf       Temp
    movfw       oneSecond
    subwf       Temp, w     ; place the result in w
    addwf       totalSecond, f
    return
```

Figure 9.2.6.3. Algorithm of getting time from RTC

9.2.7 Permanent log

The purpose of permanent log function is to keep long-term record for future reference. The registers provided in the PIC microcontroller lost its information stored when the power is cut. However, the EEPROM module would record the information even after the power is cutoff and even returned on because of its special structure. The EEPROM on the PIC16F877 chip is a 16*16 memory array, with each trunk of 2-bit hexadecimal data. The data structure we utilize the EEPROM is displayed as follows, this table represent one row of the memory space within EEPROM

0	1	2	3	4	5	6	7	8	9	A	B-F
NR	**L1**	**L2**	**L3**	**L4**	**L5**	**L6**	**L7**	**L8**	**L9**	**TS**	**N/A**

Table 1.3.7. Schematics of data stored in EEPROM

NR: Number of operation, from 0 to 15 ideally

TS: Total seconds used during operation

After one operation cycle is done, the EEPROM write module is called to write all 9 L registers data into the EEPROM. While there is a need to read from the EEPROM, the values of 9 L registers is load back to the 9 registers for the generic L register-checking module (see **Figure 9.2.5.1** for detail). The working flow is displayed below.

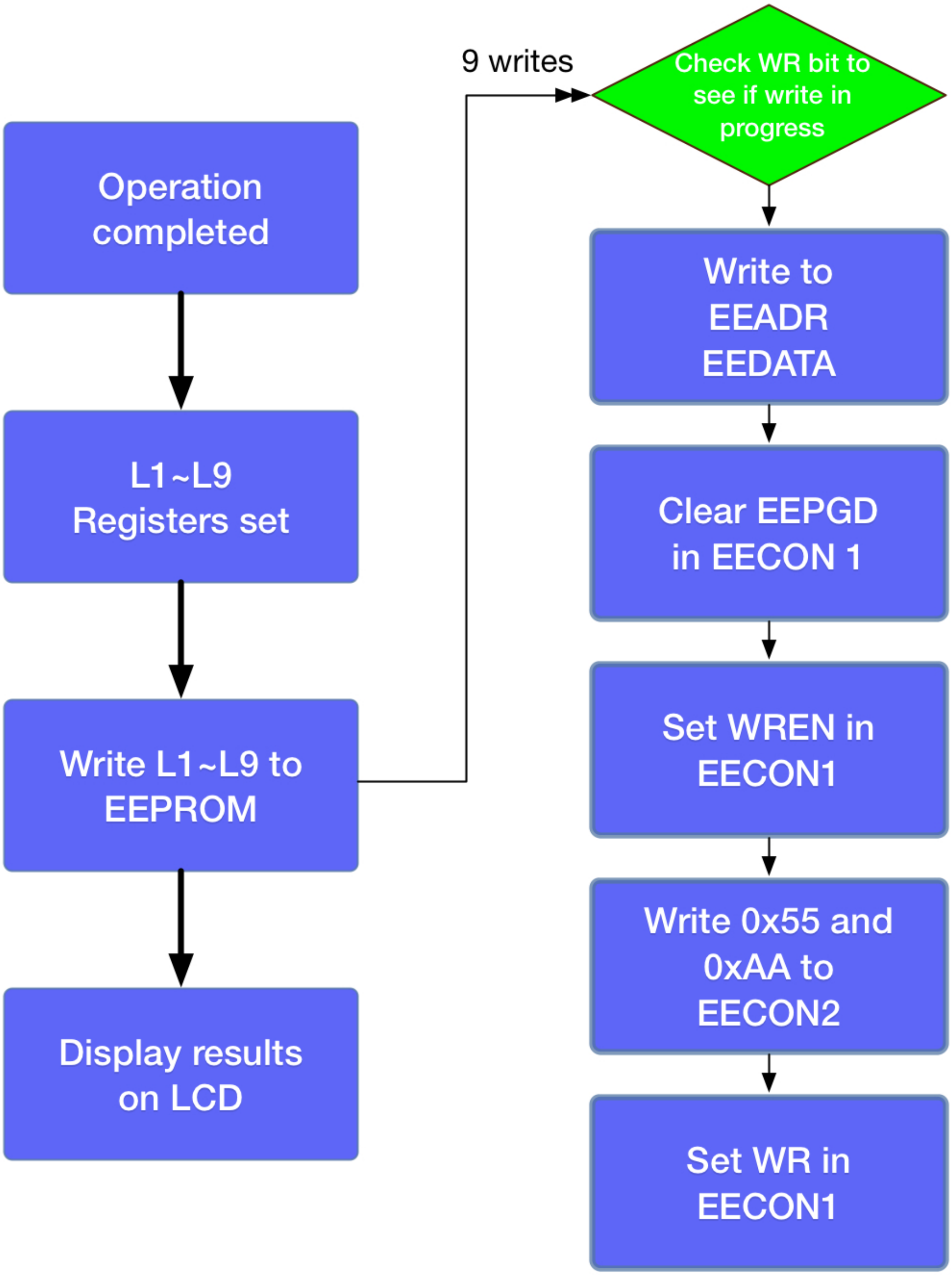


Figure 9.2.7.1. Method of writing to EEPROM

```
;****************************************
; REEPROM module
;****************************************
writeEEPROM
    ; you have to set the EADDR register yourslef in the code
    ; you have to set the EDATA by yourself

    banksel     EECON1
    bsf         STATUS, RP1
    bsf         STATUS, RP0     ; bank3
    btfsc       EECON1, WR      ; wait for write to finish
    goto        $-1

    banksel     EEADR
    bcf         STATUS, RP1     ; bank2
    movf        EADDR, W
    movwf       EEADR
    movf        EDATA, W
    movwf       EEDATA

    bsf         STATUS, RP1     ; bank3
    bcf         EECON1, EEPGD
    bsf         EECON1, WREN

    bcf         INTCON, GIE

    movlw       0x55
    movwf       EECON2
    movlw       0xAA
    movwf       EECON2

    bsf         EECON1, WR
    bsf         INTCON, GIE

    btfsc       EECON1, WR      ; wait for write to finish
    goto        $-1

    banksel     PORTA
    return
```

Figure 9.2.7.2. Code for writing to EEPROM

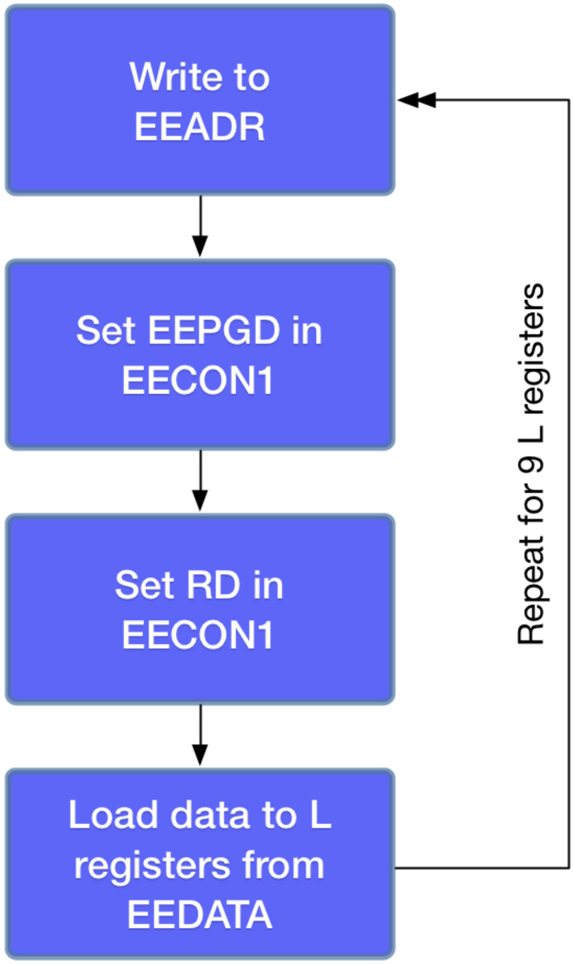


Figure 9.2.7.3. Method of reading from EEPROM

9.2.8 PC interface

The PC interface serves as a easy of recording the information on PC. IT utilizes the Universal Synchronous Asynchronous Receiver Transmitter module provided on the PIC16F877 chip. The communication occurs at a baud rate of 9600 bits per second, with 8 bits of data transferred serially through the RS232 port on the DevBugger board. HyperTerminal is required on target computer to act as the receiver as well as the decoder of the data.

9.2.9 Future Improvement

There is no perfect written code and actually there are a lot improvement can be made to the microcontroller code we have.

9.2.9.1 Division of functions in different modules

The style of code we have in this project is only one main file that contains all the code needed. However, the code is hard to debug and do maintenance to, for its unneeded complexity. Sometimes it is simply hard to look for the snippet desired in the file that has over 2000 lines of code. So the first improvement will be separate the code into different files.

Also, as can see from the code provided in the **Appendix**, every function of manipulating L registers is copied 9 times to accommodate the need for each of the registers (like the address, performance counter, etc.). Writing the functions generically can do another improvement. By saying

"Generic", one mean there will be only one function to handle the operation on 9 L registers. By assigning specific address to L registers in CBlocks, one can access the registers by referring to its address. The generic L register modules can take in that address as a parameter and directly control the register. This will reduce the code to 1/9 of its original size and improve efficiency.

9.2.9.2 Motor control

According to the design, a powerful 12 (V) driven DC motor is used to turn the lights ready for inspection on and off. The method being used right now is just assigning digital signal to the H-bridge and delay for a certain amount of time. However, due to the unpredictability of DC motors, the stop position cannot be controlled perfectly. Once the DC motor runs over position, the gears start to wear out. Improvement on this part will be introducing interrupt service accompany IR beam cut sensors. Once the IR beam is cut, the interrupt flag is raised and thus the microcontroller can stop the motor at its most optimized position. This improvement not only reduces the ware of the system but also provide better performance in terms of turning the lights on and off.

9.2.9.3 Data structure optimization

The data encoding used in the project is always one hot, ie the representation is setting only one of the bits in the however long binary string. This is good for simplifying the code checking the binary value but it is very inefficient. The reason is that in an 8-bit binary number, only 8 pieces of information can be recorded. However, if the encoding follows the conventional binary number, 256 values can be encoded. That is a big difference. In other words, the information stored in the L registers can actually be stored in only one register. Also, in the EEPROM data memory, we stored about 40 one hot coded binary number, consuming one-sixth space provided in the EEPROM. If the efficient encoding is used, we can use only 4 ~ 10 memory spots for the same amount data. The efficient storage method can significantly reduces the space needed for storing data. In later development, compared to the method that is current using, this method can extend the data space to a large extent.

10. Circuit Subsystem

10.1 Problem Assessment

Circuits behave like a linkage between the PIC microcontroller and the electromechanical. The responsible for circuity part can be summarized as:

10.1.1 Sensor System

Design light sensing circuit and presence sensing circuit, which detect external condition variations and transfer into voltage variations. The voltage variation outputs can be regarded as digital or analog signals. The output signals are sent to PIC to analyze.

10.1.2 Power Supply

Provide noise-free, stable power to all electronic pieces including sensors, actuators, and microcontrollers.

10.1.3 Actuator Driving Circuit

Force the machine parts or the candlelight switches to move dynamically.

10.1.4 Cable management and Safety Ensured

Properly manage the wires to make the machine portable, stable and no potential electric shock hazards.

10.1.5 Emergency stop

In advance, implement an emergency STOP button to cease all operations as long as the STOP button pressed. All dynamical parts will stop immediately expect the PIC, the power of PIC is independent of effect of Emergency STOP.

10.2 Solution

10.2.1 Light Sensing Circuitry:

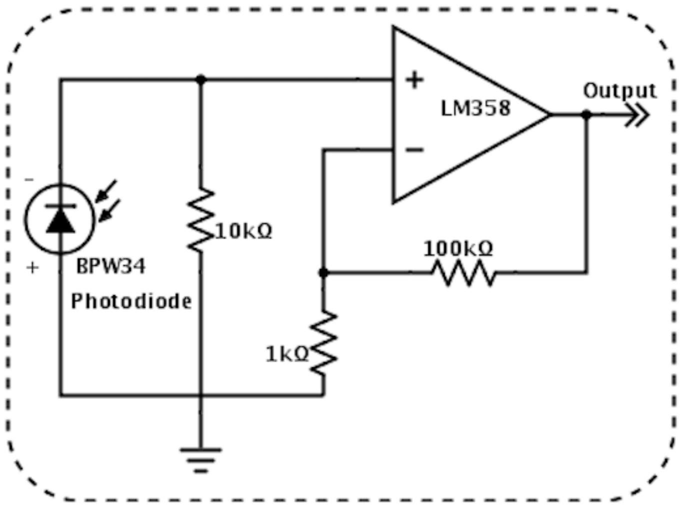


Figure 10.2.1.1. circuit (photodiode)

10.2.1.1 Functionality:

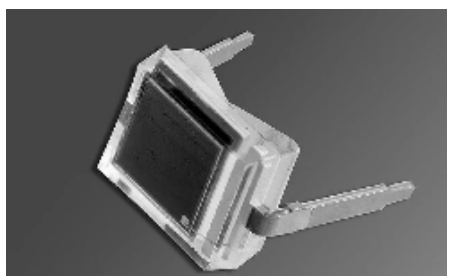


Figure 10.2.2.1. Photodiode BPW34

Light sensing circuits are powered by 5V supply.

We selected photodiode BPW34 as our sensing devices to detect candlelights intensity. Compared with the circuit for phototransistor, though price of photodiode ($1.5) is higher than phototransistor ($1.2), the photodiode circuit as a whole is much simpler and cheaper. And the shape for photodiodes is square, which is easy to immobile.

LM358 operational amplifiers were applied to magnify output voltages to an detectable magnitude for PIC. When there is a non-flicker always-on candlelight, the output voltage will be around 3.3V. If it is flickering, the voltage ranges from 2.4V to 3.2V.Light sensing circuits were fabricated and soldered according to figure

10.2.1.2 BPW34 photodiodes can detect lighting intensity:

The resistance of photodiode varies with the changing of amount of light cast on it. Locating the candlelight's wick around 0.5 centimeters from the photodiodes, when the candlelight flicker, the output voltage is fluctuating between 2.5V and 3.4V.

Also, the distance from the candlelight's wick to the photodiodes matters. If the distance is 3 centimeters or more, with constantly on candlelight, the voltage is always less than 0.6V--it is too low for PIC to recognize. If around 2 centimeters, the voltage is 1.2 ~ 2.2 (V). Only with distance less than 1 centimeter, the output voltage will be large enough. This factor is taken into consideration when we integrate the circuit with electromechanical part.

10.2.1.3 Soldering Board Section to Hold Photodiodes:

Instead of just connecting the photodiodes to jumper wires, placing them on a specific "holder" is more fixed and stable. Hence, we decided to solder them onto soldering board sections. An entire soldering board was cut apart into appropriate size by hand drill. Also, the connecting wires from the board of photodiodes to the main light sensing board are soldered.

Under the implementation of soldering board as the "holder", the relative position between the photodiode and the candlelights is identical among all 9 sensors.

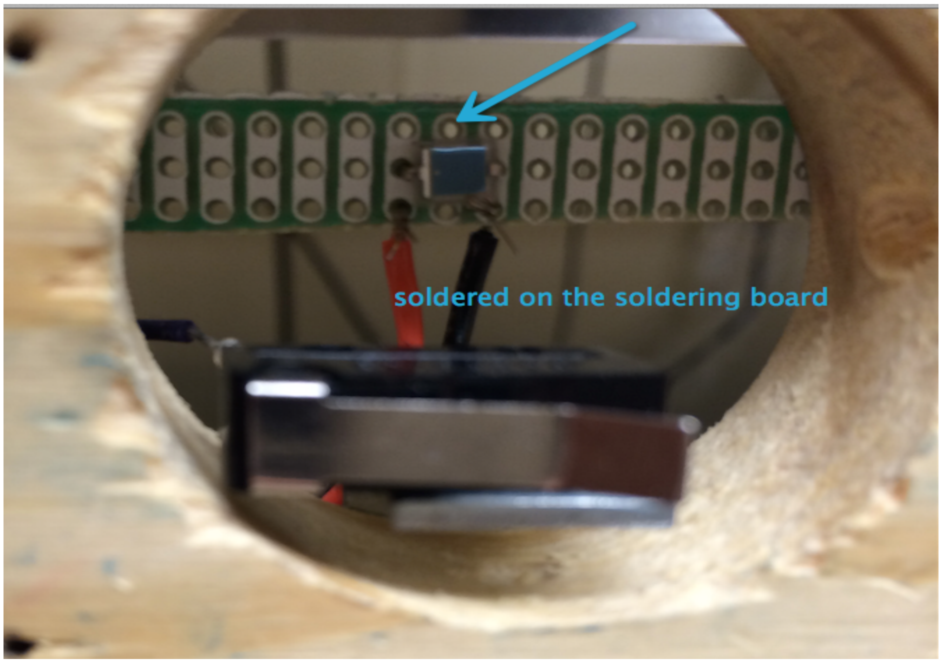


Figure 10.2.1.2. Photodiode soldered onto a soldering board section

10.2.1.4 Color Coded Wires:

In this circuit, for the wires connecting between photodiodes and main circuit board, the black wires denotes the anodes for photodiodes BPW34 which will be connected to the ground, while the red for cathode, connecting to the non-inverting input of LM358 Chip.

On the main circuit board, the white wire is supposed to connect to 5V voltage power supply and the black one is to ground. All blue wires are signal outputs from photodiodes, and they will be sent to PIC as analog signals.

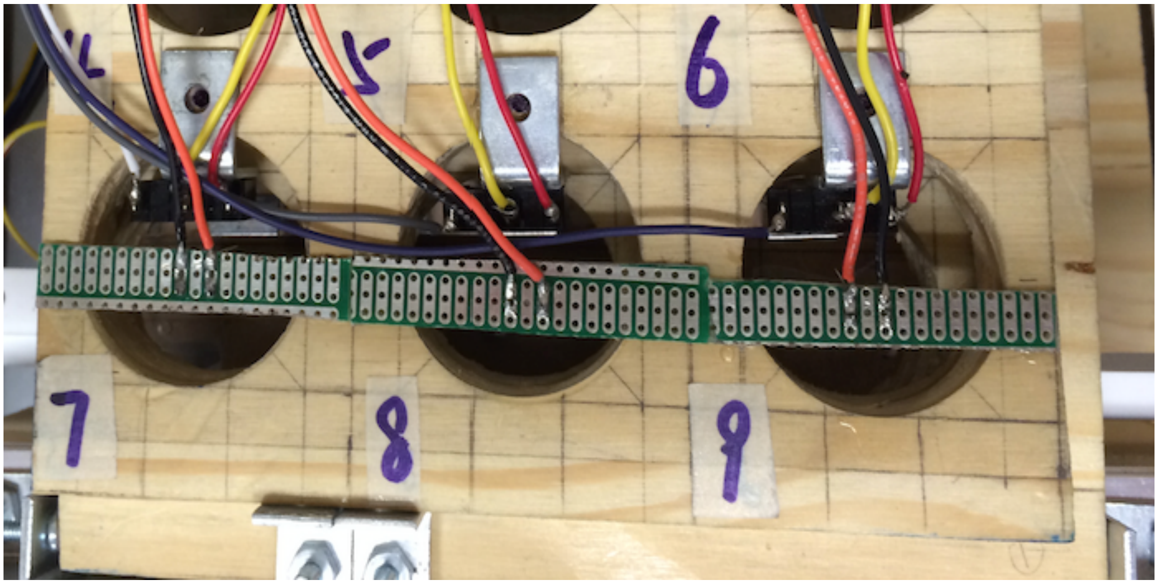


Figure 10.2.1.3. Photodiode wire color coded

10.2.2 Presence Sensing Circuity:

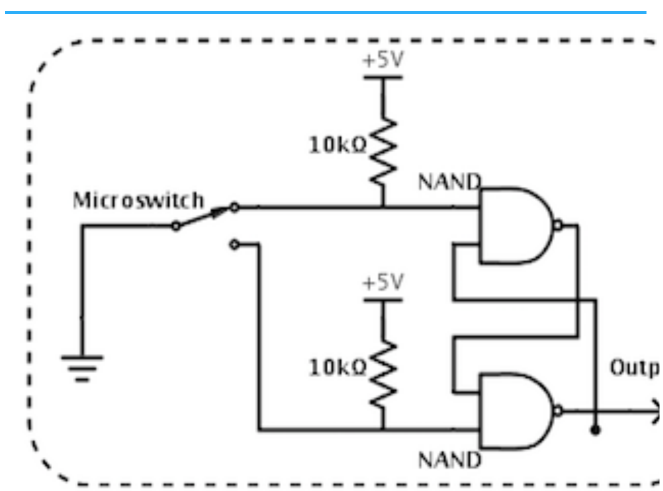


Figure 10.2.1.1. circuit (microswitch)

10.2.2.1 Functionality:

Presence sensing circuits are powered by 5V supply. Microswitches take advantages because they requires less about the accuracy of postion, compared with IR sensor, which emitter and collector must be "face-towards-face" or it will not work. Besides, higher than 4V voltage are always regarded as digital signal 1 by microcontroller. The microswitches output changes from 0V to 5V when get pressed. Implementation of NAND gates renders the output signal non-bounce.

Figure 10.2.2.1. microswitch

10.2.2.2 Color Coded Wires:

In this circuit, for the wires connecting between microswitches and main circuit board, the colorful wires is connected port 1 to ground, while yellow wires connects port 2 to 2 denoted in the microswitch circuit. Similarly, purple wires connects port 3 to 3.

On the main microswitch circuit board, the white wire is supposed to connect to 5V voltage power supply and the black one is to ground. All blue wires are signal outputs from microswitch, and they will be sent to PIC as digital signals.

10.2.3 Actuator Driving Circuit

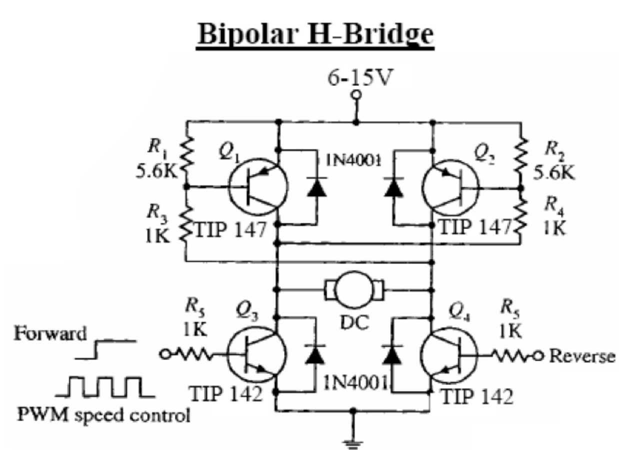


Figure 10.2.3.1. Bipolar H-Bridge Circuit

10.2.3.1 Functionality

H-Bridge circuit is powered by 12V DC voltage, and the direction control signal is given by PIC.

It is designed to actuate DC motor for candlelights switching system.

10.2.3.2 Color Coded Wires:

The yellow wires are supposed to connect to PIC. They are the controlling signals inputs. And wires colored blue are connected to poles of DC motor. According to our convention, white wire is connected to 12V +Vcc, and black one to ground.

10.2.4 Power Supply

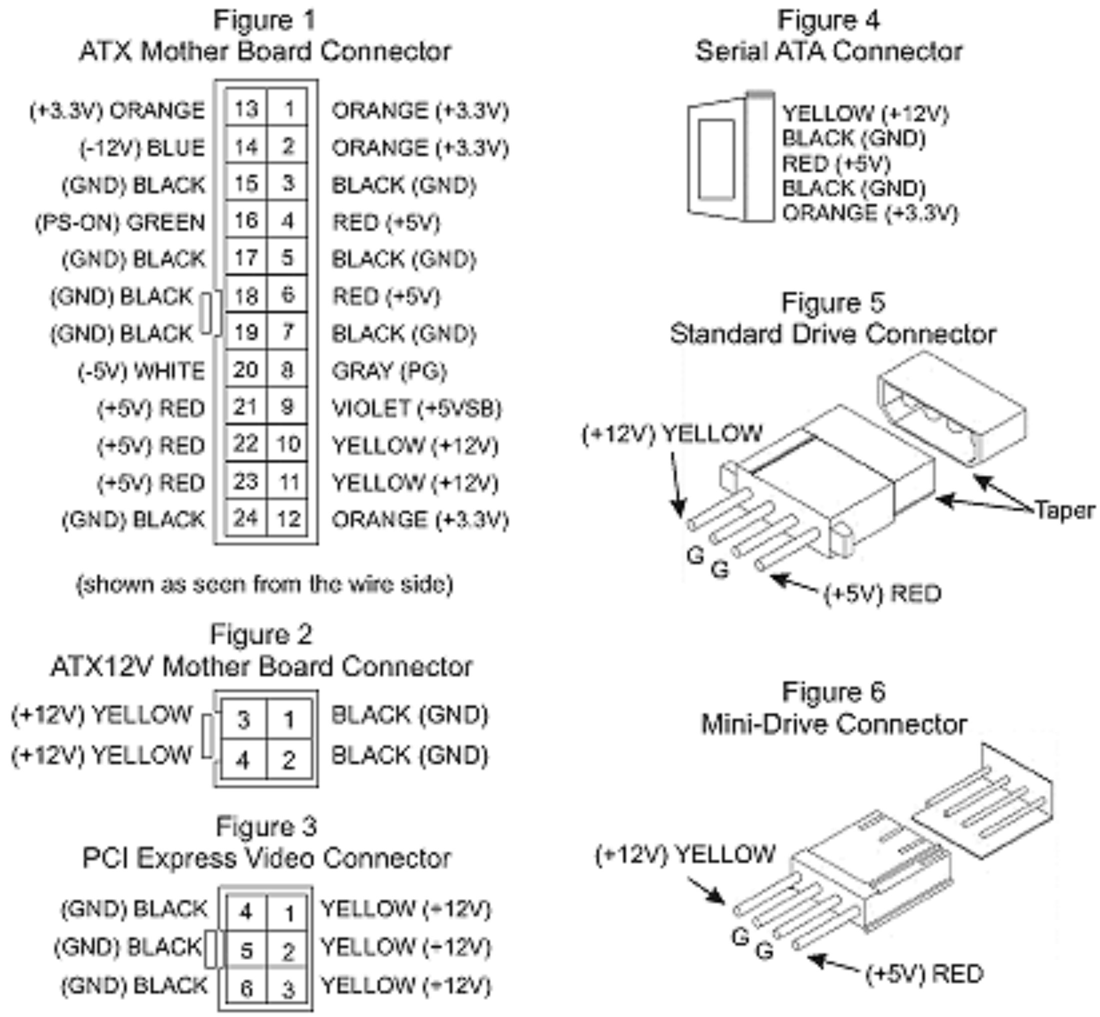


Figure 10.2.4.1. ATX12V power supply and its connector alignment

Computer power supply readily provides noise free 5V and 12V voltage supply required by all circuits and PIC Devbugger board. Besides, computer power supply have short-circuit protection feature, which immediately cuts off power when short circuit occurs. In our project, we will use ATX12V(shown in Figure) as an ideal power for all electronic units and actuators.

The output power lines of ATX12V are color coded and adequate in quantity, which makes our power connecting and debugging easier. Specifically, the yellow wires are 12V +Vcc while red ones are 5V +Vcc. Black wires indicates ground same as convention.

10.2.5 Emergency Stop and Power Line Board

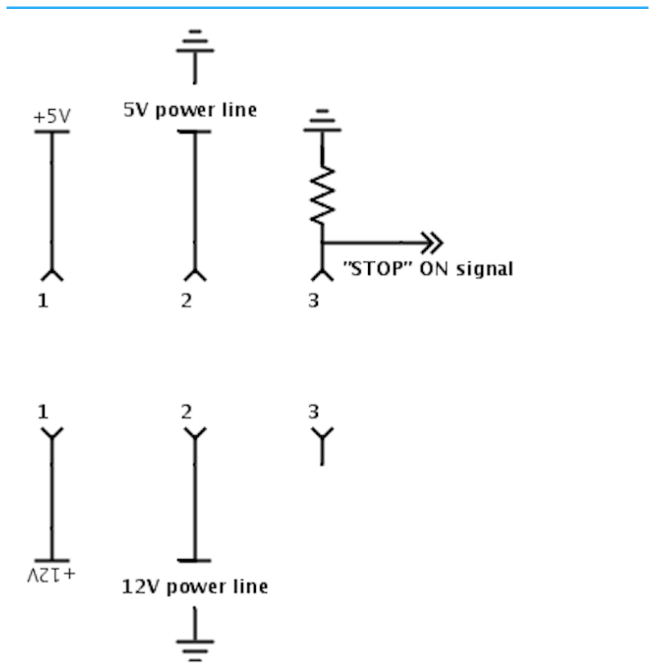


Figure 10.2.5.1. Circuit (emergency stop circuit)

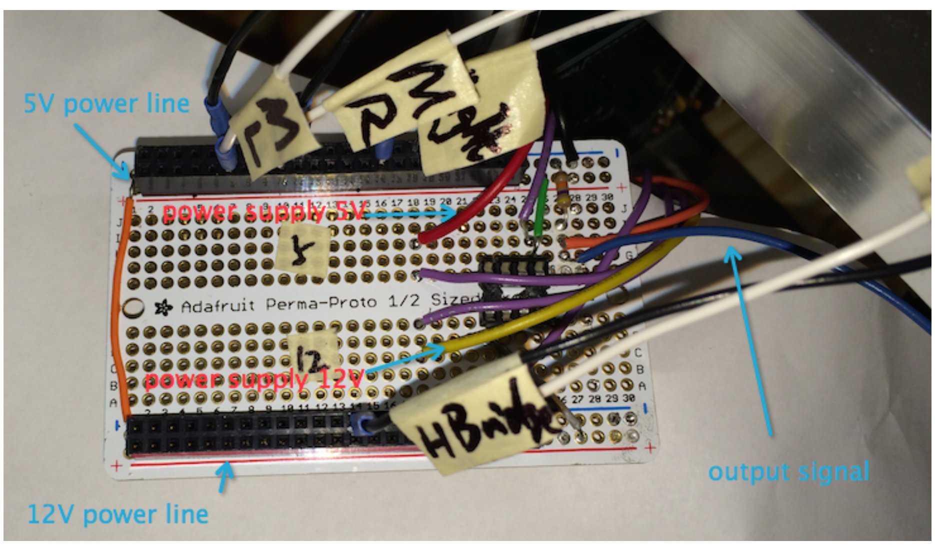


Figure 10.2.5.2. emergency stop and power board

10.2.5.1 Functionality

The purpose of power board as well as Emergency STOP board is to manage all power inputs and guarantee common ground of all circuits.Two 2×20 sockets are installed to allow power output from power board. The red line on upper part of the board is 5V power line and the lower one is 12V. Blue line denotes common ground and the orange wire connects both ground lines together.

The red line is from ATX12V power supply and it provides 5V, similarly, the yellow one is 12V. (As we noted before, the power supply output wires are color coded)Purple lines connects the board to port 1 and 2 of microswitch on both sides, and the orange lines connects to port 3.When all operations going on, meaning the emergency stop is OFF, port 1 and port 2 are connected while port 3 is open. The power outputs from power supply are connected to 5V/12V power lines to power all parts of robot except PIC. (PIC is connected to 12V from power supply directly. Therefore, even the emergency stop turns on, the PIC still works and LCD can display message such as "Emergency!")

When emergency stop on, port 1 is connected to port 3.

10.2.5.2 Emergency STOP button Selection

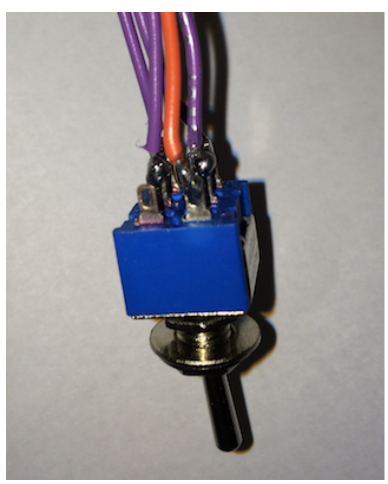


Figure 10.2.5.3. Emergency stop button

The maximum current allowance for emergency button we selected is 6A. (big enough)

As measured by multimeter, the current from two microswitch board are usually 0.12A and 0.25A respectively. And current from light sensing board are 0.23A in total. The current from H-bridge driving circuit is 0.4A usually. From all information about current indicated above, the total amount of current through the button is always less than 1A; thus the button selected always works in safety.

10.2.6 Signal Selection through implementation of multiplexer

10.2.6.1 Functionality

As PIC16 input ports are not enough for 9 analog signals and 9 digital signals from sensing Circuit plus one more digital signal input from Emergency STOP board, two CD4067BE multiplexers are applied in our circuitry. One for selecting nine digital signals from microswitch presence sensing circuit, and the other for analog signals from photodiode light sensing circuit.

Sockets are installed to facilitate the signal input for multiplexers. The port we selected in utility is 2 to 10 as labeled clearly on the side of sockets.

10.2.6.2 Color Coded Wires

Selection signals inputs are denoted by yellow and labelled. The output signals from multiplexers are both blue with tape labeling “Microswitch” or “Light”.Also, as convention, the ground is denoted by black wire while power input white wire.

10.2.7 Protect Integrated Circuit/PIC

In general, buffers are applied to current protect. It transfer a voltage from a first circuit, which has a high output impedance level, to a second circuit with a low input impedance. The interposed buffer amplifier protects the second circuit from loading the first circuit unacceptably and interfering with its desired operation.

In our robot, all voltages designed in the circuit are accorded with every electronic components data sheet and they can work safely. So for protection of IC and PIC, we focus on the current part. Corresponding to the principle of electromagnetic, if two chips connected, the current between them will negotiate by themselves; thus there is no hazard.

In our circuit, all signal outputs from sensing circuits are sent to 16-to-1 multiplexer for selection. For photodiode circuit, the output is from LM358, an operational amplifier chip. While for microswitch circuit, it is from 74HC00, a NAND logic gate. According to those, all connections are chip-to-chip. The current itself will negotiate. The working safety of circuit is guaranteed even without any buffers.For the signals sent to PIC microcontroller after selection in multiplexer, this a chip-to-chip connection, the current will be restricted in safety range. We dont need any buffers in this part.

10.2.8 Wire Management

All wires are color coded as we stated above.

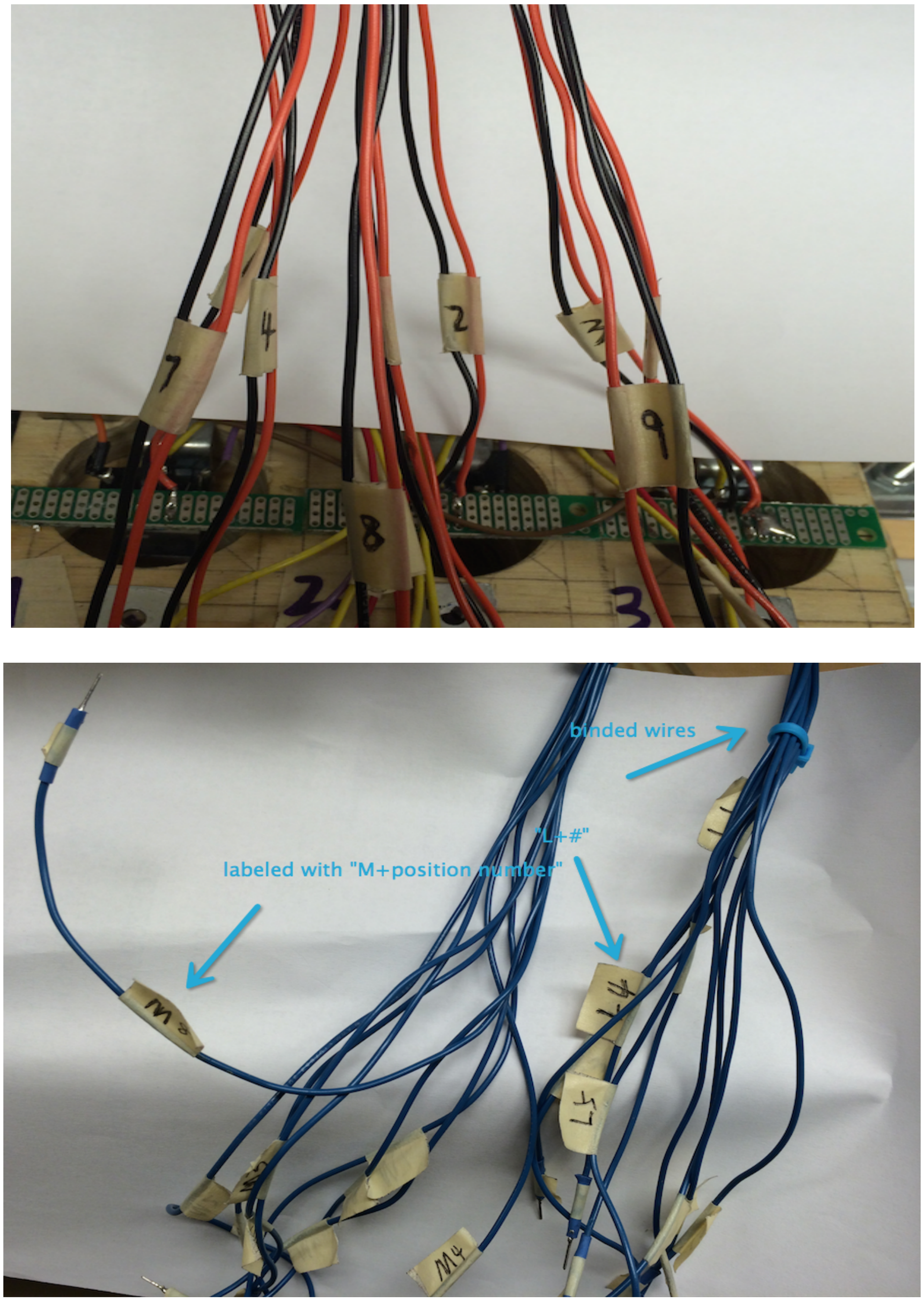


Figure 10.2.8.1. Color coded wires and labeled wires

Besides, we labeled all wires with white tapes. In particular, for blue output wires, “M” denotes microswitch, and “L” denotes light sensor. Moreover, connecting wires are tied together to make our circuit neat and tidy.

Power supply output lines are cut, stripped, and soldered to our power board (Emergency STOP board) for stability, since connecting by jumper wires are unstable and jumpers wires are tend to burn out easily when a little large current through it.

10.2.9 Suggestions for Improvement of the subsystem:

10.2.9.1 Replace H-bridge with Integrated Circuit L298N:

Taken into account economy and portability factors, it would be better if we can replace H-bridge with L298 IC. L298 is much lighter than H-bridge and also takes less room. Meanwhile, L298 IC costs less than we complete a H-bridge circuit. Fabricating a H-bridge circuit, we need a soldering board($3), 4 transistors ($1.5 each), 4 diodes (0.5 each), and some resistors (price negligible), which means at least $11 in total. Compare with L298, whose price is around $8, it is obvious the latter is a better alternative.

10.2.9.2 2.9.2 Microswitch Circuit:

We chose the microswitch circuit without bouncing when gets pressed. Compared the circuit with the one with bouncing, it needs one more NAND gate. We can simplify the circuit by replacing non-bouncing microswitch with the bouncing one. It wont influence our detect results as the bouncing time is very short and the PIC is not sensitive to that short-time variation. However, the problem of bouncing circuit is that the connection from the output of microswitch to multiplexer is no longer a chip-to-chip connection. Then buffers may be in need to provide current protection.

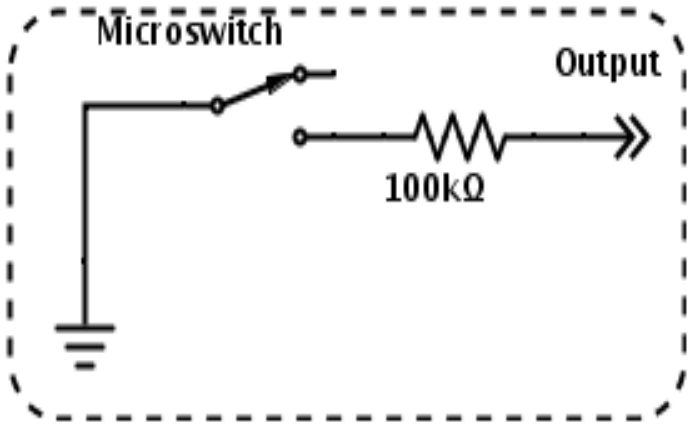


Figure 10.2.9.1. Microswitch circuit

10.2.9.3 Printed Circuit Board:

With the implementation of PCB, overall cost of circuits part can be reduced as well as the space designed for circuit board and the weight of total circuitry. Also, the connecting wires are more manageable with printed circuit board. Currently, the circuit boards design has a priority of modularity. This determines debugging and repairing relatively easy. Moreover, the PCB board has to be designed and fabricated specifically. That may be time-consuming and hard to realize since we only focus on a small project regarding machine.

10.2.9.4 Photodiode to photoresistor:

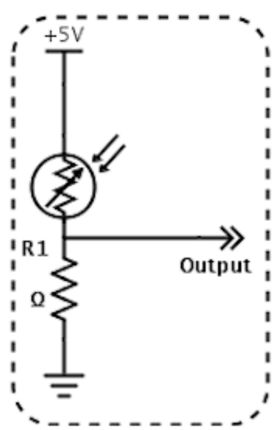


Figure 10.2.9.2. Photoresistor Cicuit

The above scheme is photoresistor circuit. Photoresistor is much simpler and lighter compared with photodiode circuit. Despite that the reaction time is longer and it works non-linearly, if the reference resistor chosen appropriate, this circuit works as well as the photodiode circuit we used in our design currently.

,

11. Integration

Integration and Calibration between Circuit and Electromechanical Components: the integration between circuits and electromechinical components includes functionally integrating photodiodes and microswitches to mechanical structures, and connecting them to circuit boards, connecting motor to actuator driving board, placing and fixing all soldering board onto the machine, and installing human interface elements such as Emergency STOP button.

11.1 Photodiode and microswitch:

Photodiodes were soldered onto soldering board sections, and then the soldering board sections are align in order, and glued onto the machine in the fixed relative position towards candlelights. Since the relative positions are nearly same for all nine candlelights, it would be easy for the microcontroller part to set reference voltage. Meanwhile, the microswitches were also glued onto the machine by super glue 502.

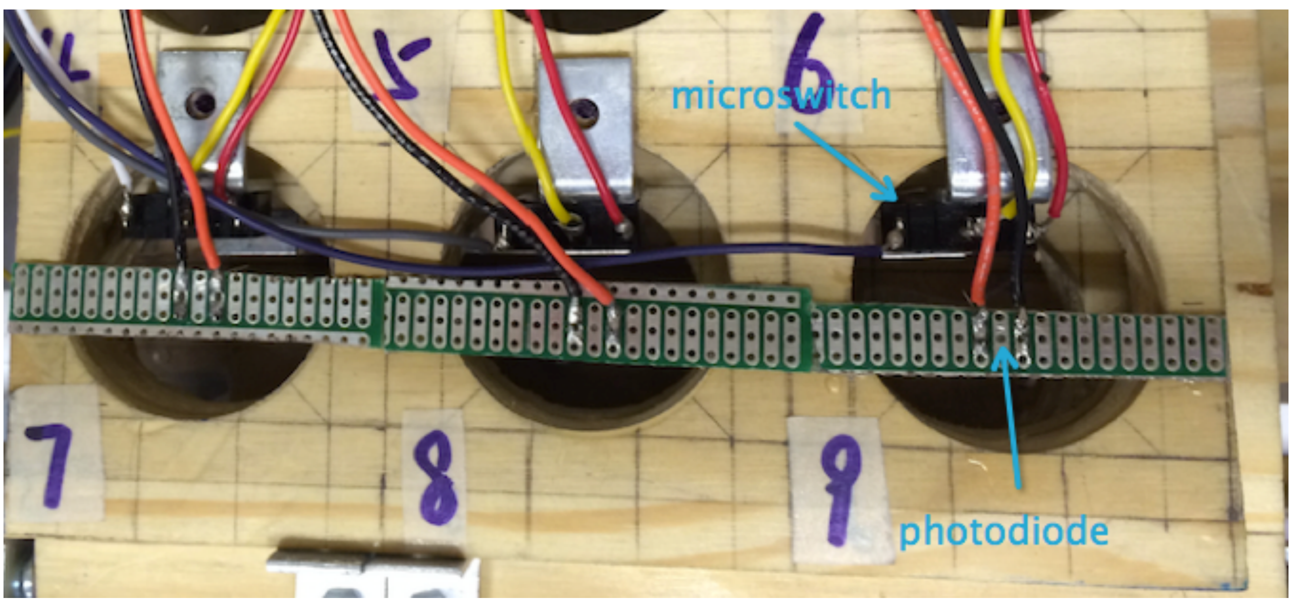


Figure. photodiode and microswitch integrated with tray cover

11.2 Soldering board allocation

All soldering boards, 5 in total as well as the Devbugger board are placed onto a transparent resin board and fixed by proper screws as a circuity panel. The circuity panel is placed at the back of machine, and it functions as a holder for all circuits as well as a shelter for closure of machine itself.

11.3 Driving circuit driving actuators:

Motors are connected to driving circuit, like H-bridge, L298 IC circuit to obtain power and control. In our case, the DC motor is connected to the H-bridge driving board on circuity panel, with PIC sending enable digital signals to H-bridge, the motor can successfully rotate forward and in the reverse direction.

11.4 Integration and Calibration between Circuit and Microcontrollers

The microcontroller board, namely the Devbugger board is powered by +5V voltage independent of emergency stop button, and it also need to be grounded. The bridge between the bus on Devbugger board and circuit boards is a 40-pin IDE cable. All controlling signals output from bus through this cable to circuits and all detecting signals are collected into the cable then send to the PIC. The main problem between microcontroller part and circuit part is about floating signals. Once a signal sent into PIC is floating, it may influence the whole performance of the PIC. Floating voltage signal is a voltage not connected by any conducting path such as resistors to ground. To avoid unexpected behavior due to floating voltage, all ports not in use are grounded.

12. Accomplished Schedule

In this section, the chronological Gantt chart proposed in the project proposal is revisited and reflected upon. As noticed in the project time line, only the electromechanical subsystem followed the preset schedule. Both microcontroller and circuit subsystem member failed to stick to the schedule, owing to the uncertainty associated in the process of integration.

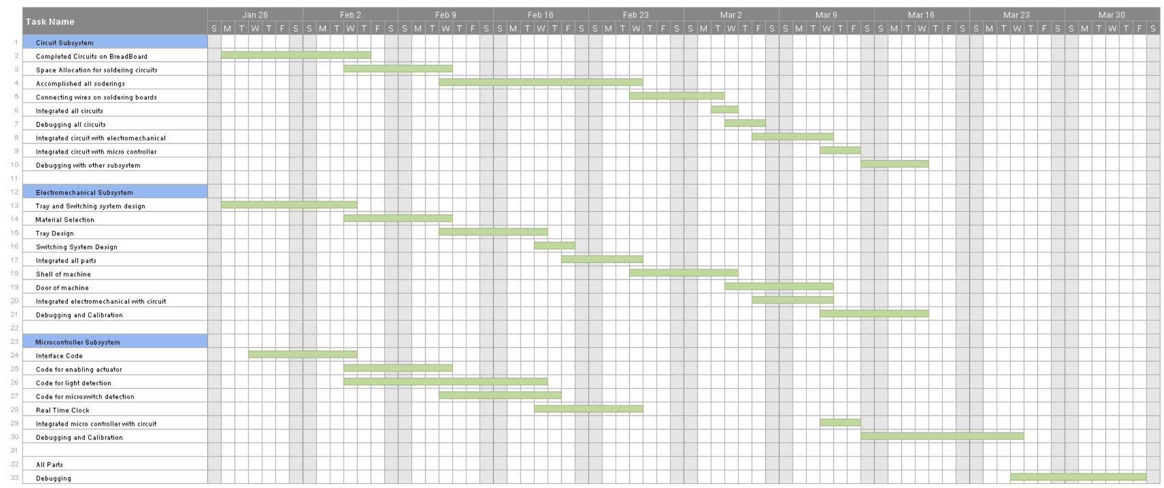


Figure 5: Project management

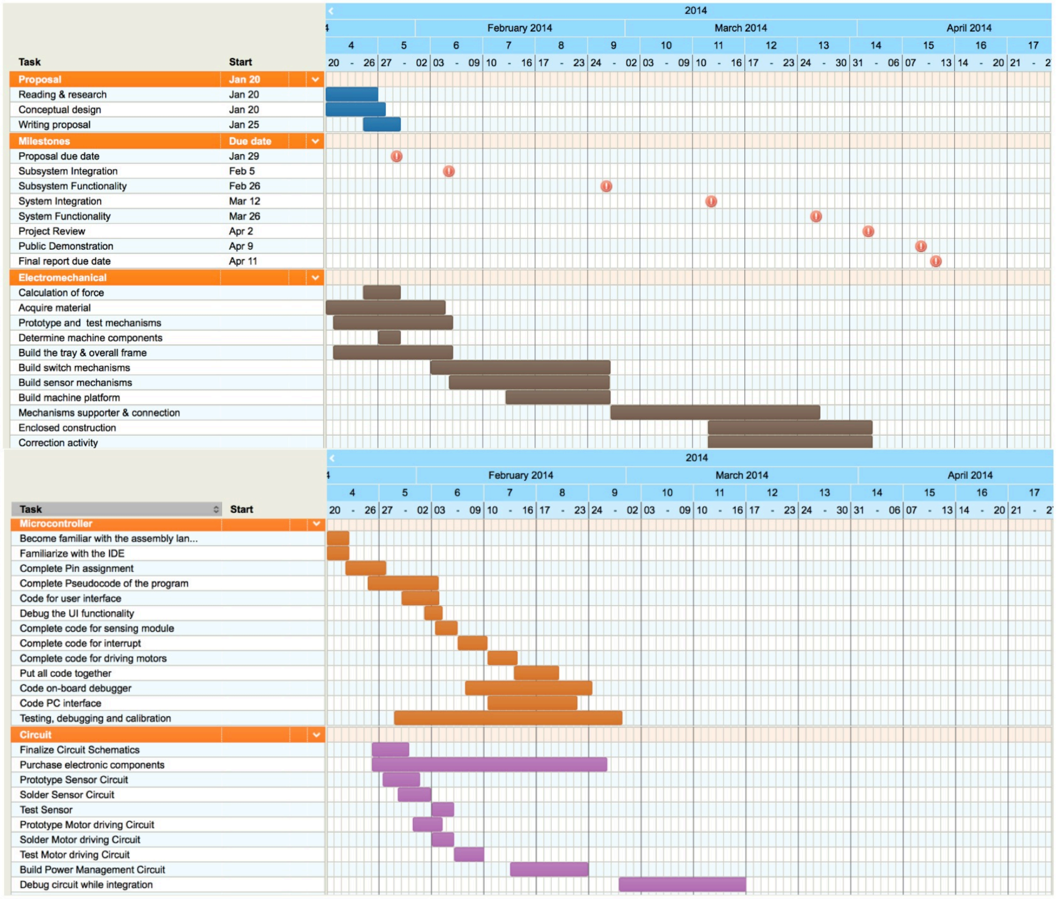


Figure 12. Project accomplished schedule (top), planned schedule (bottom)

13. Description of Overall Machine and Operating Procedure

13.1 Basic Feature

Name: Robust Robotics

Dimension: 30cm × 30cm ×40 cm

Weight: 5.5 kg

Cost: $ 222.75 Canadian dollars

Maximum Candlelights tested in a single run: nine

Operation Time: 8~9 seconds

13.2 Functionality

13.2.1 Operation

Our robot test the functionality of maximum 9 candlelights. All operation takes around 10 seconds. After the tray loaded by candlelights (it is regarded as standby mode), DC motors works to actuate the flipping switching system to open all candlelights. The microswitch goes to detect the presence of candlelights, then the total number of candlelights loaded is displayed on LCD. Afterwards, photodiode light sensing circuit works to test the functionality of existing candlelights. The spot without candlelights will be skipped to ensure efficiency. Once all detection completed, the motor moves in reverse direction to send all parts of machine including the candlelights to standby mode.

13.2.2 Tray and its cover

The three by three tray is served as a platform to hold and fix the candlelights. With the tray cover implemented, it can immobile the candlelights in particular position during operation, which means it prevents horizontal, vertical and rotary movements of candlelights. Two holes at the bottom of the tray are responsible of avoiding horizontal and rotary movements while the tray cover makes sure candlelights not shift vertically.

Microswitches and photodiodes are glued onto the cover of the tray by super glue.

13.2.3 Candlelights Switch System

The switch system of our robot is reliable and simple, also it can avoid jam caused by friction forces during operations. Three parts-- a moveable base plate, mental drawer slide and rack with pinion, compose it. Mental drawer slide works with rack and pinion, and they are driven by motor to remove the moveable base plate forth and back. With the movement of base plate, the wood blocks stuck to the base plate encounter the switches to force them on/off.

13.2.4 Real Time Clock

It can display current time and date in second accuracy on LCD screen and even external power supply cut off, it still works and maintains in right time and date.

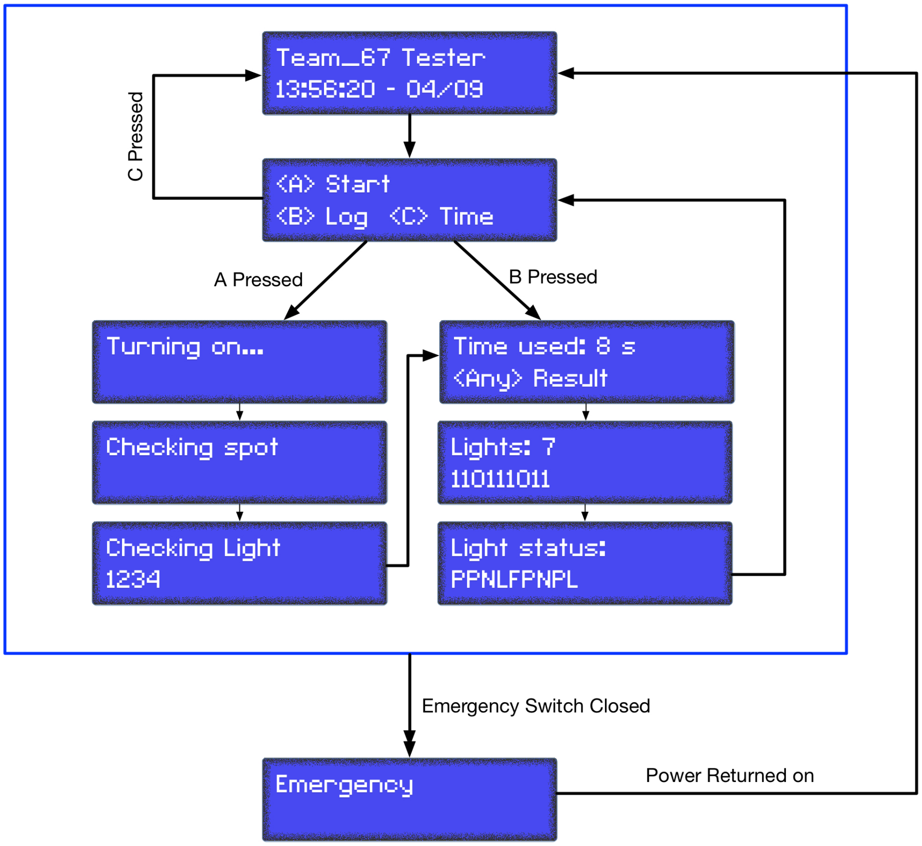
13.2.5 2.5 Notification of Emergency STOP ON

When the Emergency STOP on, meaning all operations cease immediately. The message "Emergency!" will be sent to PIC and displayed on LCD screen.

Figure 13.1. Display on LCD screen when emergency stop ON

13.3 3.Operating Procedure:

13.3.1 3.1 Operation Messages on LCD screen



The above figures indicate all instructions on LCD and machine working flow.

13.3.2 3.2 Before the operation

The machine is normally in standby mode. Load maximum 9 candlelights onto the tray, make sure their positions fixed the holes at the bottom of the tray. Put the tray back to machine, and lock the tray.

Load the candlelights as the picture below instructed:

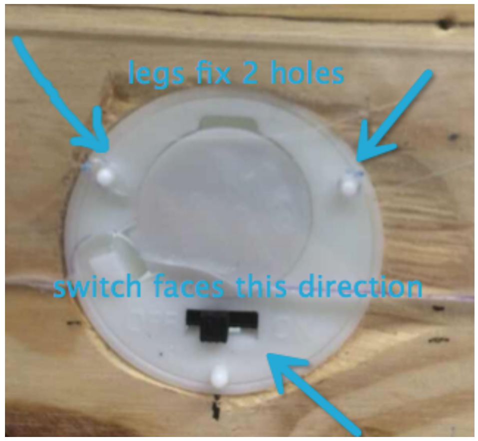


Figure 13.2 . Instruction for candlelights loading

13.3.3 3.3 In process

All operations are autonomous once we start the machine. The entire operation is normally completed within 10 seconds. Upon completion of the operation, a termination message will be displayed on the LCD screen.

3.4 After Operation

After all operation, the machine are back to “stand-by” condition, we can unload candlelights from the machine and read the results from LED screen on the left side.

First, the screen displays the total number of candlelights aligned on the tray.

After press any key to continue, it goes to display the testing results in order.

“N” means no candlelights there;

“F” means flicker failure, candlelights are constantly on;

“P” means pass, candlelights flicker;

“L” means LED failure, candlelights are constantly off.

For example, “FLPPNNLPF” on LCD screen means candlelights at port 1 and 9 are constantly on; while ones at port 2 and 7 are constantly off; Candlelights at port 3, 4, 8 functions well while there is no candlelights loaded at port 5 and 6.

14. System Issues and Improvements:

14.1 Portability

The machine we designed weighs 5.5 kilograms and dimension 30*30*40 (in centimeter). It complies with constraints but still too heavy and cumbersome. In order to make it more portable, some lighter material can be chosen to fabricate the frame of our machine and shell of it. In particular, it is higher practical to replace all resin board we used with firm cardboard. In addition, installing wheels underneath the machine is doable. Besides, we have found a smaller and lighter, also cheapter power supply online. It can took place of the one which we currently used.

14.2 Toggle Emergency Button SHOULD be changed to pressed one

As mentioned by our TA, push button, compared with toggle one, would be expected to implementing emergency stop. Because if something urgent and really dangerous happens, pressing a button on is more assessable than toggling one. Taken into account the purpose of emergency stop button, the pushed one is more suitable.

Figure . Push Button in desire

14.3 PIC18 replacs PIC 16:

Without installation of multiplexer, there will be 19 input signals, of which 9 are analog from photodiode circuit, 9 are digital signals from microswitch circuit and the rest one are digital signal from emergency stop board. Besides, 2 output digital signals are intended to enable and control the H-bridge and the DC motor.

PIC 16 only have eight analog ports, which is not enough without the help of multiplexer. While PIC 18 can be applied to provide enough analog and digital ports with multiplexer required.

Employment of multiplexer helps lower machine budget, reduce the weight of machine as soldering board is rather heavy compared with other components. In addition, owing to the chosen multiplexer is 16 to 1, there are some ports leaving unused, which suggests they need to be grounded to avoid floating voltage effect. This means a lot of extra work and harden debugging. After all, PIC 18 is more suitable than PIC 16 for our design machine.

15. Conclusion

"Robust Robotics" is an autonomous candlelight-testing machine, which was designed and constructed over the past four months. This machine is capable of testing the functionality of at most 9 candlelights in a single run, opening the switches of candlelights, testing, after that turning off the candlelights and returning to the original state. It satisfies all the constraint and requirements stated previously and it can operates reliably throughout the testing process.

The construction of design can be categorized into three: fabrication of the tray, fabrication of switching system, assembly of the sensing system and motor system, and the closure mechanism. Candlelights are loaded onto the tray and once operation starts, switching system driven by DC motor turns on all candlelights except those failing LED (constantly off). Once the candlelights on, the sensing system works to detect the presence of candlelights and distinguish their functionality. The result of each candlelight will be sent to PIC microcontroller one by one. As long as all nine ports information collected, the switching system continues to work, while the motor rotates in reverse direction to turn off all candlelights.

The design performs consistently and it performed two high-qualified runs during public demonstration. It successfully test eight out of nine candlelights functionality except the one loaded at port 5 and displays results onto LED screen after operation accomplished. The total operation time for one run is around 9 seconds, which is under the limitation of 90 seconds.

In addition, "Robust Robotics" also has extra design features, such as real time clock, closure of the design, and it enjoys the potential of extendibility to more than nine candlelights testing. Nevertheless, the design has some issues and some improvement can be made to optimize its performance and functionality. Firstly, the toggle emergency stop button should be changed to push one in order to optimize its functionality when emergency happens. Secondly, the material used to closing all machine can be reselected considering that the machine is not light enough and it is possible to make it more portable. Last but not least, replacing PIC 16 with PIC 18 is practical in order to reduce the number of circuits.

16. Reference

[1] M.Reza Emami “Multidiscipline Engineering Design from Theory to Practice”, University of Toronto, 2014.

[2] R. Emami, "Aer 201 Lecture Slides," 2014. [Online]. Available: http://sps.aerospace.utoronto.ca/courses/aer201/Pages/LectureNotes.aspx

[3] Digikey, “Digikey,” 2014. [Online]. Available: http://www.digikey.ca/

[4] "Most useless machine upgrade," 7 August 2011 . [Online]. Available:http://hackaday.com/2011/08/07/most-useless-machine-upgrade-now-with-a-button/.

[5] T. Fischer, "Roulette Barrel Gun," 29 November 2011. [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/b/bb/Relais_Animation.gif. [Accessed 28 Jan 2014].

[6] Creatron Inc., Lilypad Light Sensor. [Online] available: http://www.creatroninc.com/index.php/lilyp-398464.html

[7] [Online]. Available: http://upload.wikimedia.org/wikipedia/commons/b/bb/Relais_Animation.gif. [Accessed 28 Jan 2014].

[8] Darmawaskita HArtono, Measure light intensity with an 8-bit microcontroller. [Online] available: http://electronicdesign.com/lighting/measure-light-intensity-8-bit-microcontroller

[9] Makeblock, Switch Test Machine by using makeblock. [Online] available: https://www.youtube.com/watch?v=35s8QObG7bo

[10] RACK AND PINION GEAR SYSTEMS. [Online] available: http://www.technologystudent.com/gears1/gears4.htm

[11] Rotational Inertia. [Online] available: http://physics.info/rotational-inertia/

[12] WHAT IS TORQUE. [Online] available:
http://www.physics.uoguelph.ca/tutorials/torque/Q.torque.intro.html

[13] From Radio Receivers to LED Flashlights: An LED Odyssey. [Online] available: http://www.coastportland.com/blog/receivers-to-led-flashlights-led-odyssey-infographic/

17. Appendix

Assembly code for microcontroller

```
          list p=16f877                  ; list directive to
define processor
      #include <p16f877.inc>         ; processor specific
variable definitions
      #include <rtc_macros.inc>
      __CONFIG _CP_OFF & _WDT_OFF & _BODEN_ON & _PWRTE_ON &
_HS_OSC & _WRT_ENABLE_ON & _CPD_OFF & _LVP_OFF

    ; vaiables for RS232 module
    offset  EQU       0x20
    temp    EQU       0x21
    sel_0   EQU       b'0000'
    sel_1   EQU       b'1000'
    sel_2   EQU       b'0100'
    sel_3   EQU       b'1100'
    sel_4   EQU       b'0010'
    sel_5   EQU       b'1010'
    sel_6   EQU       b'0110'
    sel_7   EQU       b'0111'
    sel_8   EQU       b'0001'


    cblock  0x22
        ; variables to store data
        l1
        l2
        l3
        l4
        l5
        l6
        l7
        l8
        l9

        COUNTH
        COUNTM
        COUNTL
        Table_Counter
        lcd_tmp
        lcd_d1
        lcd_d2
        com
```

```
        dat
        shiftCount
        delayCount
        Temp
        Time
        ; variable for timer
        minute
        tenSecond
        oneSecond
        ; store intermidiate value
        totalSecond
        dig1
        dig10
        LED_Counter

        ; variables for adcon
        Threshold   ; for the light sensor
        Counter     ; for identifying the behavior
        Loop        ; count the looping
        LED         ; for display infomation on LCD

        ; MUX
        mux

        ; EEPROM code
        EADDR
        EDATA
        turn

    endc


    ;Declare constants for pin assignments (LCD on PORTD)
        #define RS   PORTD,2
        #define E    PORTD,3

;***************************************
; Display macro
;***************************************
Display macro   Message
        local   loop_
        local   end_
        clrf    Table_Counter
        clrw
loop_   movf    Table_Counter,W
        call    Message
```

```
        xorlw   B'00000000' ;check WORK reg to see if 0 is
returned
        btfsc   STATUS,Z
        goto    end_
        call    WR_DATA
        incf    Table_Counter,F
        goto    loop_
end_
        endm

tableAccess macro ENTRY
    bcf STATUS ,C
    movwf Temp
    movlw  HIGH ENTRY
    movwf   PCLATH
    movf  Temp, w
    addlw   LOW ENTRY
    btfsc STATUS ,C
        incf PCLATH ,f
    movwf PCL
    endm


         ORG        0x0000     ;RESET vector must always be
at 0x00
         goto       init       ;Just jump to the main code
section.

         ORG         0x0004      ; the interupt code

interupt
    ; the interupt set when rb0 set to high on emergency
    banksel     INTCON

    btfss   INTCON, INTF         ; the interupt comes from
rb0 changepic1
    goto    $+4
    bcf     INTCON, INTF         ; clear the flag
    goto    displayEmergency     ; and fall into a loop
until another interupt is called
    nop
    btfss   PORTB, 0             ; if the portb is set,
meaning emergency, display message
    goto    $-2
    ; go back work
    goto    DisplayOptions
```

```
displayEmergency
    Call        Clear_Display
    Display     Emergency_msg
    nop
    return

;***************************************
; Delay: ~160us macro
;***************************************
LCD_DELAY macro
    movlw   0xFF
    movwf   lcd_d1
    decfsz  lcd_d1,f
    goto    $-1
    endm

;***************************************
; helper functions
;***************************************
shiftLeft macro count
    movlw   count
    movwf   shiftCount
    call    shiftLeft_0
    endm

shiftLeft_0
    Call    HalfS
    movlw   b'00011000'
    call    WR_INS
    decfsz  shiftCount, f
    goto    $-4
    return

shiftRight macro count
    movlw   count
    movwf   shiftCount
    call    shiftRight_0
    endm

shiftRight_0
    Call    HalfS
    movlw   b'00011100'
    call    WR_INS
    decfsz  shiftCount, f
    goto    $-4
```

```
    return

delayForTimeInterval macro time ; time unit: 0.5s
    movlw   time
    movwf   delayCount
    call    delay
    endm

delay
    call    HalfS
    decfsz  delayCount, f
    goto    $-2
    return

BCD_Converter
    CLRF    dig1
    movwf   dig1
    clrf    dig10
    movlw   0xA
    subwf   dig1, F
    btfss   STATUS, C
    goto    Divide_Done
    incf    dig10
    goto    $-4

Divide_Done
    addwf   dig1, F
    movlw   0x30
    addwf   dig10, F
    addwf   dig1, F
    return



;***************************************
; DisplayRS macro
;***************************************
DisplayRS macro Message
        ;
        ; the macrofunction to display table message on the
screen
        ; modified from Display
        ;
        local   loopRS_
        local   endRS_
```

```
        clrf    Table_Counter
        clrw

loopRS_
        movf    Table_Counter,W
        call    Message
        xorlw   B'00000000' ;check WORK reg to see if 0 is
returned
        btfsc   STATUS,Z
        goto    endRS_
        call    RS_Write
        incf    Table_Counter,F
        goto    loopRS_

endRS_
        endm



;***************************************
; Initialize LCD
;***************************************
init
         ;; setting up the interupt
         banksel        INTCON
         bsf            INTCON, GIE
         bsf            INTCON, INTE    ; enable rb0
interrupt

         ;; remember to set b0 as input


         ; +Summury for Pin out
         ; +Last update, Apr9, 1138 UTC
         ; |
         ; +A: 8 ports
         ; ||- RA0:5 set to ADC function
         ; ||- All 6 are input, pass through ADC
         ; ||- TODO:
         ; |
         ; +B: 8 ports
         ; ||- RB7:4, RB1: keyboard input
         ; ||- RB0,2,3 -----> Interupt function
         ; ||- TODO: N/A
         ; |
         ; +C: 8 ports
```

```
        ; ||- RC0: light number check
        ; ||- RC3:4: RTC
        ; ||- RC5:7, 2:1: Free ->  RC7,6,2,1 for mux1
        ; ||- TODO:
        ; |
        ; +D: 8 ports
        ; ||- RD7:2: LCD
        ; ||- RD1:0: Motor
        ; |
        ; +E: 3 ports
        ; ||- Not used

        banksel   PORTA
        movlw     b'00000010'    ; set up the adcon1
        movwf     ADCON1
        clrf      PORTA
        clrf      PORTB
        clrf      PORTC
        clrf      PORTD

        banksel   TRISA


        movlw     b'000111'
        movwf     TRISA
        movlw     b'11111111'    ; Set required keypad
inputs
        movwf     TRISB


        movlw     b'00100001'
        movwf     TRISC


        clrf      TRISD          ; All port D is output
        clrf      TRISE
        bsf       TRISE, 1

        ; initaiate the l* variables
        call      setInitialValueForLightResigters

        banksel    PORTA
        call       initRTC       ;Initialize the LCD (code
in lcd.asm; imported by lcd.inc)
        call       InitLCD
        call       initRS
```

```
         goto       Main

;***************************************
; Main code
;***************************************
Main
        Call        Clear_Display
        Display     Welcome_Msg
        delayForTimeInterval 4

DisplayOptions
        call        Clear_Display
        Display     Menu_1
        call        Switch_Lines
        Display     Menu_2

        ;shiftLeft   h'03'
        movlw       b'00000010'
        call        WR_INS

KeyInputDetection
         btfss      PORTB,1     ;Wait until data is
available from the keypad
         goto       $-1
         btfsc      PORTB,1     ;Wait until key is released
         goto       $-1
         btfsc      PORTB,7
         goto       C_PRESSED
         btfsc      PORTB,6
         goto       B_PRESSED
         btfsc      PORTB,5
         goto       A_PRESSED
         goto       KeyInputDetection

A_PRESSED
        btfss       PORTB,4
        goto        KeyInputDetection

        call        setInitialValueForLightResigters

        ; read the RTC initial input
        rtc_read    0x01
        movfw       0x78
        movwf       minute
        rtc_read    0x00
```

```
        movfw       0x77
        movwf       tenSecond
        movfw       0x78
        movwf       oneSecond

        ;;;;;;;;;;;;;;;;;;;;;
        ; motor control             ; PORTD,0 ff, PORTD,1
rev
        ;;;;;;;;;;;;;;;;;;;;;

        call        Clear_Display
        Display     Motor_fwd_msg

        call        Motor_On_Forward
        ;delayForTimeInterval 6
        call        Motor_Off

        ;goto        DisplayOptions

        ;;;;;;;;;;;;;;;;;;;;;
        ; check the spot
        ;;;;;;;;;;;;;;;;;;;;;
        call        Clear_Display
        Display     Spoting_Msg
        call        Switch_Lines

        call        checkMicroswitch ; it gets all spot
checking done!
        call        Clear_Display
        ;;;;;;;;;;;;;;;;;;;;;
        ; check the number
        ;;;;;;;;;;;;;;;;;;;;;
        call        Check_number

        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        ; check the functionality
        ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
        call        Clear_Display

        Display     Light_Checking_msg
        call        Switch_Lines

        movlw       "1"
        call WR_DATA
        call        selectMux_1
        call        Set_Value
```

```
        movlw      b'01010100'       ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf      Threshold

        btfss       l1, 0

        call        Convert_AD
        call        regLight_1

        movlw       "2"
        call WR_DATA
        call        selectMux_2
        call        Set_Value

        movlw      b'01010010'       ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf      Threshold

        btfss       l2, 0

        call        Convert_AD
        call        regLight_2

        movlw       "3"
        call WR_DATA
        call        selectMux_3
        call        Set_Value

        movlw      b'01011000'       ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf      Threshold

        btfss       l3, 0

        call        Convert_AD
        call        regLight_3

        movlw       "4"
        call WR_DATA
        call        selectMux_4
        call        Set_Value
```

```
        movlw      b'01011000'       ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf      Threshold

        btfss       l4, 0

        call        Convert_AD
        call        regLight_4

        movlw       "5"
        call WR_DATA
        call        selectMux_5
        call        Set_Value

        movlw      b'01111000'       ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf      Threshold
;
        btfss       l5, 0

        call        Convert_AD
        call        regLight_5

        movlw       "6"
        call WR_DATA
        call        selectMux_6
        call        Set_Value

        movlw      b'01011000'       ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf      Threshold

        btfss       l6, 0

        call        Convert_AD
        call        regLight_6

        movlw       "7"
        call WR_DATA
        call        selectMux_7

        call        Set_Value
```

```
        movlw       b'01010000'         ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf       Threshold
;
        btfss        l7, 0
        call         Convert_AD
        call         regLight_7

        movlw        "8"
        call WR_DATA
        call         selectMux_8
        call         selectMux_8
        call         Set_Value

        movlw       b'01011000'         ;5*153/255 = 3v is the
reference value for the light sensor threshold ; use 192
3.75v
        movwf       Threshold

        btfss        l8, 0
        call         Convert_AD
        call         regLight_8

        movlw        "9"
        call WR_DATA
        call         selectMux_9
        call         Set_Value

        movlw       b'01011000'
        movwf       Threshold

        btfss        l9, 0
        call         Convert_AD
        call         regLight_9

        ; turn the lights back off
        call         Clear_Display
        Display      Motor_rev_msg

        call         Motor_On_Reverse
        call         shortDelay
        call         shortDelay
        call         Motor_Off
```

```
        ; time calculation
        call        checkTimeElapsed    ; time stored in
dig10, dig1
        movfw       totalSecond
        call        BCD_Converter
        call        Clear_Display


        call        Clear_Display
        ;Display     Done_Msg
        Display     Time_Done_Msg   ; display the "time
used: %d s"
        movfw       dig10
        call        WR_DATA
        movfw       dig1
        call        WR_DATA
        Display     Time_Done2_Msg
        call        Switch_Lines
        Display     Done_2
        goto        RESULTS

B_PRESSED

        movfw       totalSecond
        call        BCD_Converter
        call        Clear_Display
        ;Display     Done_Msg
        Display     Time_Done_Msg   ; display the "time
used: %d s"
        movfw       dig10
        call        WR_DATA
        movfw       dig1
        call        WR_DATA
        Display     Time_Done2_Msg
        call        Switch_Lines
        Display     Done_2
    call        RESULTS

C_PRESSED
        btfsc       PORTB,6
        goto        KeyInputDetection
        btfss       PORTB,5
        goto        KeyInputDetection
        btfss       PORTB,4
        goto        KeyInputDetection
```

```
        goto            show_RTC


Motor_On_Forward
        bsf             PORTD, 0
        call            shortDelay
        ;call            shortDelay
        call            Motor_Off
        return
Motor_On_Reverse
        bsf             PORTD, 1
        return
Motor_Off    ; tune PORTD, 0
        bcf             PORTD, 0
        return

Check_number
        banksel         PORTA

        clrf            LED_Counter
        btfss           l1, 0    ; skip if the 0 bit of l1 is
set, add counter otherwise
        incf            LED_Counter
        btfss           l2, 0    ; skip if the 0 bit of l2 is
set, add counter otherwise
        incf            LED_Counter
        btfss           l3, 0    ; skip if the 0 bit of l3 is
set, add counter otherwise
        incf            LED_Counter
        btfss           l4, 0    ; skip if the 0 bit of l4 is
set, add counter otherwise
        incf            LED_Counter
        btfss           l5, 0    ; skip if the 0 bit of l5 is
set, add counter otherwise
        incf            LED_Counter
        btfss           l6, 0    ; skip if the 0 bit of l6 is
set, add counter otherwise
        incf            LED_Counter
        btfss           l7, 0    ; skip if the 0 bit of l7 is
set, add counter otherwise
        incf            LED_Counter
        btfss           l8, 0    ; skip if the 0 bit of l8 is
set, add counter otherwise
        incf            LED_Counter
```

```
        btfss       l9, 0   ; skip if the 0 bit of l9 is
set, add counter otherwise
        incf        LED_Counter
        return




Check_function
                call                Convert_AD
                Display         Status_Msg
                movfw               LED
                call                WR_DATA
                return


RESULTS
        ; ?check??
        call        Check_number

        btfss       PORTB,1     ;Wait until jsondata is
available from the keypad
        goto        $-1
        btfsc       PORTB,1     ;Wait until key is released
        goto        $-1
        call        Clear_Display
        ;call        Switch_Lines

        ;display the number of lights       ; P57 on
notebook
        Display     Quantity
        movfw       LED_Counter
        call        BCD_Converter
        movfw       dig1
        call        WR_DATA

        ;; display specfic data
;        call        Switch_Lines
;        call        logMicroswitch
        ;; delay
;        delayForTimeInterval    6

        btfss       PORTB,1     ;Wait until jsondata is
available from the keypad
        goto        $-1
        btfsc       PORTB,1     ;Wait until key is released
```

```
        goto        $-1
        call        Clear_Display

        ; display behavior
        call        Clear_Display
        Display     Status_Msg
        call        Switch_Lines
        call        logLights

        btfss       PORTB,1     ;Wait until jsondata is
available from the keypad
        goto        $-1
        btfsc       PORTB,1     ;Wait until key is released
        goto        $-1
        call        Clear_Display
        ;;
        ;; testing the rs code here
        ;;
        ;DisplayRS   Status_Msg

;        delayForTimeInterval    6
        call        Clear_Display

        goto        DisplayOptions

;***************************************
; Look up table
;***************************************
Welcome_Msg
    tableAccess Welcome_log
Welcome_log
    dt  "Team 67_tester", 0
Menu_1
    tableAccess Menu_log_2
Menu_log_2
    dt  "<A> Start", 0
Menu_2
    tableAccess Menu_log
Menu_log
    dt  "<B> Log <C>:Time", 0

Spoting_Msg
    tableAccess Spoting_log
Spoting_log
    dt  "Checking Spot", 0
```

```
Time_Msg
    tableAccess Time_log
Time_log
    dt  "Return in 1 seconds", 0

Time_Done_Msg
    tableAccess Time_Done_log
Time_Done_log
    dt  "Time used:  ", 0

Time_Done2_Msg
    tableAccess Time_Done2_log
Time_Done2_log
    dt  "s", 0

Status_Msg
    tableAccess Status_log
Status_log
    dt  "Light status: ", 0

Done_2
    tableAccess Done_2_log
Done_2_log
    dt  "<Any key> Result", 0

Quantity
    tableAccess QualityLog
QualityLog
    dt  "Lights: ", 0

Quality_light_good
    tableAccess QualityGoodLightLog
QualityGoodLightLog
    dt  "Flashlight1:", 0

Light_Checking_msg
    tableAccess Light_Checking_log
Light_Checking_log
    dt  "Checking Light ", 0

Motor_fwd_msg
    tableAccess Motor_fwd_log
Motor_fwd_log
    dt  "Turning on", 0

Motor_rev_msg
```

```
    tableAccess Motor_rev_log
Motor_rev_log
    dt  "Turning off", 0

Emergency_msg
    tableAccess Emergency_log
Emergency_log
    dt  "Emergency!", 0

Reset_msg
    tableAccess Reset_log
Reset_log
    dt  "Reset!", 0

;***************************************
; LCD control
;***************************************
Switch_Lines
        movlw   B'11000000'
        call    WR_INS
        return

Clear_Display
        movlw   B'00000001'
        call    WR_INS
        return

;***************************************
; Delay 0.5s
;***************************************
HalfS
    local   HalfS_0
      movlw 0x88
      movwf COUNTH
      movlw 0xBD
      movwf COUNTM
      movlw 0x03
      movwf COUNTL


HalfS_0
      decfsz COUNTH, f
      goto   $+2
      decfsz COUNTM, f
      goto   $+2
      decfsz COUNTL, f
```

```
      goto   HalfS_0

      goto $+1
      nop
      nop
        return

shortDelay
    local   shortDelay_0
      movlw 0x28
      movwf COUNTH
      movlw 0x5
      movwf COUNTM
      movlw 0x3
      movwf COUNTL


shortDelay_0
      decfsz COUNTH, f
      goto   $+2
      decfsz COUNTM, f
      goto   $+2
      decfsz COUNTL, f
      goto   shortDelay_0

      goto $+1
      nop
      nop
        return

AD_Delay
    local   AD_Delay_0
      movlw 0x88
      movwf COUNTH
      movlw 0x0C
      movwf COUNTM
      movlw 0x01          ;12d38,77112  1E1F  11a0
      movwf COUNTL


AD_Delay_0
      decfsz COUNTH, f
      goto   $+2
      decfsz COUNTM, f
      goto   $+2
      decfsz COUNTL, f
```

```
      goto   AD_Delay_0

      goto $+1
      nop
      nop
        return

OneS
        local   OneS_0
      movlw 0x10
      movwf COUNTH
      movlw 0x7A
      movwf COUNTM
      movlw 0x06
      movwf COUNTL

OneS_0
    banksel     PORTB
      decfsz COUNTH, f
      goto   $+2
      decfsz COUNTM, f
      goto   $+2
      decfsz COUNTL, f
      goto   OneS_0

      goto $+1
      nop
      nop
        return


;******* LCD-related subroutines *******


    ;***********************************
InitLCD
    bcf STATUS,RP0
    bsf E     ;E default high

    ;Wait for LCD POR to finish (~15ms)
    call lcdLongDelay
    call lcdLongDelay
    call lcdLongDelay

    ;Ensure 8-bit mode first (no way to immediately
guarantee 4-bit mode)
```

```
    ; -> Send b'0011' 3 times
    movlw   b'00110011'
    call    WR_INS
    call lcdLongDelay
    call lcdLongDelay
    movlw   b'00110010'
    call    WR_INS
    call lcdLongDelay
    call lcdLongDelay

    ; 4 bits, 2 lines, 5x7 dots
    movlw   b'00101000'
    call    WR_INS
    call lcdLongDelay
    call lcdLongDelay

    ; display on/off
    movlw   b'00001100'
    call    WR_INS
    call lcdLongDelay
    call lcdLongDelay

    ; Entry mode
    movlw   b'00000110'
    call    WR_INS
    call lcdLongDelay
    call lcdLongDelay

    ; Clear ram
    movlw   b'00000001'
    call    WR_INS
    call lcdLongDelay
    call lcdLongDelay
    return
    ;************************************

    ;ClrLCD: Clear the LCD display
ClrLCD
    movlw   B'00000001'
    call    WR_INS
    return

    ;****************************************
    ; Write command to LCD - Input : W , output : -
    ;****************************************
WR_INS
```

```
    bcf     RS              ;clear RS
    movwf   com             ;W --> com
    andlw   0xF0            ;mask 4 bits MSB w = X0
    movwf   PORTD           ;Send 4 bits MSB
    bsf     E               ;
    call    lcdLongDelay    ;__    __
    bcf     E                           ;  |__|
    swapf   com,w
    andlw   0xF0            ;1111 0010
    movwf   PORTD           ;send 4 bits LSB
    bsf     E                           ;
    call    lcdLongDelay    ;__    __
    bcf     E                           ;  |__|
    call    lcdLongDelay
    return

    ;****************************************
    ; Write data to LCD - Input : W , output : -
    ;****************************************
WR_DATA
    bsf     RS
    movwf   dat
    movf    dat,w
    andlw   0xF0
    addlw   4
    movwf   PORTD
    bsf     E               ;
    call    lcdLongDelay    ;__    __
    bcf     E                           ;  |__|
    swapf   dat,w
    andlw   0xF0
    addlw   4
    movwf   PORTD
    bsf     E               ;
    call    lcdLongDelay    ;__    __
    bcf     E                           ;  |__|
    return

lcdLongDelay
    movlw h'20'
    movwf lcd_d2
LLD_LOOP
    LCD_DELAY
    decfsz lcd_d2,f
    goto LLD_LOOP
    return
```

```
;*************************************
; Initilization of both LCD and RTC
;*************************************
initRTC

         ;Set SDA and SCL to high-Z first as required for
I2C

         bsf       STATUS,RP0
         bsf       TRISC,4
                 bsf       TRISC,3

         bcf       STATUS,RP0     ; select bank 0


         ;Set up I2C for communication
         call      i2c_common_setup
         ;rtc_resetAll

         ;Used to set up time in RTC, load to the PIC when
RTC is used for the first time
         ;call     set_rtc_time
         return
show_RTC

        call    Clear_Display
        ;Get year
        movlw   "2"             ;First line shows
20**/**/**
        call    WR_DATA
        movlw   "0"
        call    WR_DATA
        rtc_read    0x06        ;Read Address 0x06 from
DS1307---year
        movfw   0x77
        call    WR_DATA
        movfw   0x78
        call    WR_DATA

        movlw   "/"
        call    WR_DATA

        ;Get month
        rtc_read    0x05        ;Read Address 0x05 from
DS1307---month
```

```
        movfw   0x77
        call    WR_DATA
        movfw   0x78
        call    WR_DATA

        movlw   "/"
        call    WR_DATA

        ;Get day
        rtc_read    0x04        ;Read Address 0x04 from
DS1307---day
        movfw   0x77
        call    WR_DATA
        movfw   0x78
        call    WR_DATA

        movlw   B'11000000'     ;Next line displays
(hour):(min):(sec) **:**:**
        call    WR_INS

        ;Get hour
        rtc_read    0x02        ;Read Address 0x02 from
DS1307---hour
        movfw   0x77
        call    WR_DATA
        movfw   0x78
        call    WR_DATA
        movlw           ":"
        call    WR_DATA

        ;Get minute
        rtc_read    0x01        ;Read Address 0x01 from
DS1307---min
        movfw   0x77
        call    WR_DATA
        movfw   0x78
        call    WR_DATA
        movlw           ":"
        call    WR_DATA

        ;Get seconds
        rtc_read    0x00        ;Read Address 0x00 from
DS1307---seconds
        movfw   0x77
        call    WR_DATA
        movfw   0x78
```

```
        call    WR_DATA


        call    OneS            ;Delay for exactly one
seconds and read DS1307 again

        goto    DisplayOptions

;***************************************
; Setup RTC with time defined by user
;***************************************
set_rtc_time

        rtc_resetAll    ;reset rtc

        rtc_set 0x00,   B'10000000'

        ;set time
        rtc_set 0x06,   B'00010100'     ; Year
        rtc_set 0x05,   B'00000100'     ; Month
        rtc_set 0x04,   B'00001000'     ; Date
        rtc_set 0x03,   B'00000010'     ; Day
        rtc_set 0x02,   B'00000100'     ; Hours
        rtc_set 0x01,   B'01010101'     ; Minutes
        rtc_set 0x00,   B'00110000'     ; Seconds
        return




;***************************************
; Delay 0.01s
;***************************************
ShortDelay
    local   ShortDelay_0
      movlw 0x88
      movwf COUNTH
      movlw 0x0B
      movwf COUNTM
      movlw 0x00
      movwf COUNTL


ShortDelay_0
      decfsz COUNTH, f
      goto   $+2
```

```
        decfsz COUNTM, f
        goto   $+2
        decfsz COUNTL, f
        goto   HalfS_0

        goto $+1
        nop
        nop
          return

;***************************************
; time calculation
;***************************************
multiplyByTen
    movwf   Temp
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    movfw   Temp
    return

multiplyBySix
    movwf   Temp
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    addwf   Temp, 1
    movfw   Temp
    return


checkTimeElapsed
    ; first check the minute by subtract them
    clrf    totalSecond     ; init the register
    ;Get minute

    rtc_read     0x01       ;Read Address 0x01 from DS1307-
--min
    movfw        0x78        ; minute right now
```

```
    movwf       Temp
    movfw       minute      ; minute read when starting in
W
    subwf       Temp, w     ; place the result in w
    call        multiplyByTen
    call        multiplyBySix       ; time the minute by
60 -> second in W
    movwf       totalSecond

    ; calculate second
    rtc_read     0x00         ; read the seconds
    movfw       0x77        ; 10digit to w
    movwf       Temp
    movfw       tenSecond
    subwf       Temp, w     ; place the result in w
    call        multiplyByTen
    addwf       totalSecond, f

    movfw       0x78        ; 10digit to w
    movwf       Temp
    movfw       oneSecond
    subwf       Temp, w     ; place the result in w
    addwf       totalSecond, f
    return

;****************************************
; Analog digital converting
;****************************************
Convert_AD
        banksel     ADCON0
        call        Set_Value
        movlw       b'11010001'     ; use RA2
        movwf       ADCON0
        call        AD_CONV
;        ;movlw       b'00000000'
;        ;xorwf       ADRESH, W
;        btfss       ADRESH, 7   ; if the msb of adresh is
0, test the 6th digit
;        btfss       ADRESH, 6   ; if the 6th digit is zero
;        ;btfss       ADRESH, 5   ; if the 5th digit is
nonzero, set 1
;        ;btfss       STATUS, Z
;        goto        $+4
;        movlw       "0" ; light off or empty
;        movwf       LED
;        goto        ENDLP
```

```
    ;;;;;;;;;;
        movlw       b'11000111'         ;
        xorwf       Counter, W
        btfss       STATUS, Z
        goto        $+4
        movlw       b'0010'    ;"2" ; constant light on
        movwf       LED
        goto        $+10; goto nop
        movlw       b'00000001' ;
        xorwf       Counter, W
        btfss       STATUS, Z
        goto        $+4
        movlw       b'0100' ;    "1" ;constant light off
        movwf       LED
        goto        $+3; goto nop
        movlw       b'1000'     ;"3" ;flickering
        movwf       LED
        ;goto        ENDLP
        nop
        banksel     PORTA
        return

;ENDLP
;        movfw       LED
;        call        WR_DATA
;        goto        ENDLP

AD_CONV
        bsf         ADCON0, GO      ;start conversion and
wait for it to complete
        btfsc       ADCON0, GO
        goto        $-1
        movf        ADRESH, W
        subwf       Threshold, W    ;x/5*1023
        btfss       STATUS, C       ;C = 0, threshold > w
        goto        $+2
        goto        $+3
        decf        Counter, F
        goto        $+2
        incf        Counter, F
        call        AD_Delay          ;HalfS
        movlw       b'01100010'             ; should be 49
        xorwf       Loop, W
        btfss       STATUS, Z
        goto        $+2
        goto        $+3
```

```
        incf         Loop, F
        goto         AD_CONV
        return

Set_Value
        movlw       b'01100100'       ;counter start from 50
and increament or decreament
        movwf       Counter
        ;;
        ; the reference is set to individual light
        ;;
        movlw       b'00000000'
        movwf       Loop
        movlw       b'0000'
        movwf        LED
        return

;****************************************
; RS232 PC interface
;****************************************
initRS
        bsf         STATUS,RP0      ; select bank 1
        clrf        TRISD

        ;Setup USART for RS232
        movlw       d'15'           ; BAUD rate 9600, assuming
10MHz oscillator
        movwf       SPBRG
        clrf        TXSTA           ; 8 bits data ,no,1 stop

        bcf         STATUS,RP0      ; select bank 0
        bsf         RCSTA,SPEN      ; Asynchronous serial port
enable

        bsf         STATUS,RP0      ; select bank 1

        return
;********** Send welcome message **********************
        bcf         STATUS,RP0      ; select bank 0
        clrf        offset          ; Reset offset to 0


RS_Write
            ; move anything into W, single charactor and
log it onto the screen
```

```
            goto      $
        movwf     TXREG           ; Send contents of W to
RS232

        bsf       STATUS,RP0      ; Go to bank with TXSTA
        btfss     TXSTA,1         ; check TRMT bit in TXSTA
(FSR) until TRMT=1
        goto      $-1
        bcf       STATUS,RP0      ; Go back to bank 0

        return



;********* Table containing welcome message
********************
TAB
    tableAccess TAB_log
TAB_log
    dt  "Team 67_tester", 0

;***************************************
; Integrating function
;***************************************

; TODO: 1. Code for microswitch checking
; TODO: 2. Light sensor -----> 3cm: 2v
; TODO: 3. Logging

; LOGGING
setInitialValueForLightResigters
    movlw       b'0001'         ; LSB: 0001 as initial value:
constant off light
    movwf       l1
    movwf       l2
    movwf       l3
    movwf       l4
    movwf       l5
    movwf       l6
    movwf       l7
    movwf       l8
    movwf       l9

    clrf        Threshold

    return
```

```
;*************************************
; Mux control
;*************************************

selectMux
    ; select the input port using the signal provided in W
    ; selectionSignal: 1001
    ; RB023, RC5 as selecion -> RC5, RB3, RB2, RB0
    ; TODO: select RC as output, in init section

    ; setting the selection
    ; if the selectionSignal = 0000, needed to clear the
bit first

;    bcf       STATUS,RP0
;    bsf       STATUS,RP1    ; select bank 1
    banksel     TRISC

    bcf         TRISC, 7
    bcf         TRISC, 6
    bcf         TRISC, 2
    bcf         TRISC, 1

;    bcf       STATUS,RP0
;    bcf       STATUS,RP1    ; select bank 0
    banksel     PORTC



    bcf        PORTC, 7
    bcf        PORTC, 6
    bcf        PORTC, 2
    bcf        PORTC, 1
    ;bcf        PORTB, 2
    ;bcf        PORTB, 0
    ;delayForTimeInterval 4

    btfsc      mux, 3
    bsf        PORTC, 7
    btfsc      mux, 2
    bsf        PORTC, 6
    btfsc      mux, 1
    bsf        PORTC, 2
    btfsc      mux, 0
    bsf        PORTC, 1
```

```
    return

selectMux_1
    movlw       b'0100'
    movwf       mux
    call        selectMux
    return

selectMux_2
    movlw       b'1100'
    movwf       mux
    call        selectMux
    return

selectMux_3
    movlw       b'0010'
    movwf       mux
    call        selectMux
    return

selectMux_4
    movlw       b'1010'
    movwf       mux
    call        selectMux
    return

selectMux_5
    movlw       b'0110'
    movwf       mux
    call        selectMux
    return

selectMux_6
    movlw       b'1110'
    movwf       mux
    call        selectMux
    return

selectMux_7
    movlw       b'0001'
    movwf       mux
    call        selectMux
    return

selectMux_8
```

```
    movlw       b'1001'
    movwf       mux
    call        selectMux
    return

selectMux_9
    movlw       b'0101'
    movwf       mux
    call        selectMux
    return


;***************************************
; Check microswitches
;***************************************
checkMicroswitch
    ; RC0 is for checking the presence of the led lights
    ; by switching the selection signal of the mux, read
the port
    ; register the status on l* registers
    ; 0001  no light
    ; 0000  wait for further inspection of function


    banksel     TRISC
    bsf         TRISC, 0
    banksel     TRISB
    bsf         TRISB, 2
    banksel     PORTC



;    bcf         PORTC, 0
     check spot 1
    movlw       b'0000'
    movwf       mux
    call        selectMux
    call        selectMux_1
    movlw       "1"
    call WR_DATA
    call        AD_Delay;shortDelay
    btfsc       PORTC,0         ; skip if the port is not
set, proceed if the port is set
    bcf         l1, 0           ; clear the 0 digit of the
l1 register
```

```
    ; check spot 2
    bcf         PORTC, 0        ; clear portc to prevent
interference
    movlw       b'1000'
    movwf       mux
    call        selectMux
    call        selectMux_2
    movlw       "2"
    call WR_DATA
    call        AD_Delay;shortDelay

    btfsc       PORTC,0         ; skip if the port is not
set, proceed if the port is set
    bcf         l2, 0           ; clear the 0 digit of the
l1 register

    ; check spot 3
    bcf         PORTC, 0
    movlw       b'0100'
    movwf       mux
    call        selectMux
    call        selectMux_3
    movlw       "3"
    call WR_DATA
    call        AD_Delay;shortDelay

    btfsc       PORTC,0         ; skip if the port is not
set, proceed if the port is set
    bcf         l3, 0           ; clear the 0 digit of the
l1 register


    ; check spot 4
    bcf         PORTC, 0
    movlw       b'1100'
    movwf       mux
    call        selectMux
    call        selectMux_4
    movlw       "4"
    call WR_DATA
    call        AD_Delay;shortDelay

    btfsc       PORTC,0         ; skip if the port is not
set, proceed if the port is set
```

```
    bcf          l4, 0          ; clear the 0 digit of the
l1 register


    ; check spot 5
    bcf          PORTC, 0
    movlw        b'0010'
    movwf        mux
    call         selectMux
    call         selectMux_5
    movlw        "5"
    call WR_DATA
    call         AD_Delay;shortDelay

    btfsc        PORTC,0        ; skip if the port is not
set, proceed if the port is set
    bcf          l5, 0          ; clear the 0 digit of the
l1 register

    banksel      PORTB
    ; check spot 6
    bcf          PORTC, 0
    movlw        b'1010'
    movwf        mux
    call         selectMux
    call          selectMux_1    ;selectMux_6
    movlw        "6"
    call WR_DATA
    call         AD_Delay;shortDelay

    btfsc        PORTB, 3       ; skip if the port is not
set, proceed if the port is set
    bcf          l6, 0          ; clear the 0 digit of the
l1 register


    ; check spot 7
    bcf          PORTC, 0
    movlw        b'0110'
    movwf        mux
    call         selectMux
    call          selectMux_2 ;selectMux_7
    movlw        "7"
    call WR_DATA
    call         AD_Delay;shortDelay
```

```
    btfsc       PORTB, 3          ; skip if the port is not
set, proceed if the port is set
    bcf         l7, 0            ; clear the 0 digit of the
l1 register


     ;check spot 8
    bcf         PORTC, 0
    movlw       b'0111'
    movwf       mux
    call        selectMux
    call         selectMux_3     ;selectMux_8
    movlw       "8"
    call WR_DATA
    call        AD_Delay;shortDelay

    btfsc       PORTB, 3         ; skip if the port is not
set, proceed if the port is set
    bcf         l8, 0            ; clear the 0 digit of the
l1 register


    ; check spot 9
    bcf         PORTC, 0
    movlw       b'0001'
    movwf       mux
    call        selectMux
    call         selectMux_4     ;selectMux_9
    movlw       "9"
    call WR_DATA
    call        AD_Delay;shortDelay

    btfsc       PORTB, 3          ; skip if the port is not
set, proceed if the port is set
    bcf         l9, 0            ; clear the 0 digit of the
l1 register


    ; needed to clear up the mux selection
    movlw       b'0000'
    movwf       mux
    call        selectMux
    call        selectMux_1

    call        Clear_Display
    return
```

```
logMicroswitch
    btfsc       l1, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l1, 0
    call        write1
    nop
    btfsc       l2, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l2, 0
    call        write1
    btfsc       l3, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l3, 0
    call        write1
    nop
    btfsc       l4, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l4, 0
    call        write1
    nop
    btfsc       l5, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l5, 0
    call        write1
    nop
    btfsc       l6, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l6, 0
    call        write1
    nop
    btfsc       l7, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l7, 0
    call        write1
    nop
    btfsc       l8, 0       ; skip if l1,0 is cleared
(present),
    call        write0
```

```
    btfss       l8, 0
    call        write1
    nop
    btfsc       l9, 0       ; skip if l1,0 is cleared
(present),
    call        write0
    btfss       l9, 0
    call        write1
    nop
    return

write1
    movlw       "1"
    call        WR_DATA
    return

write0
    movlw       "0"
    call        WR_DATA
    return

;***************************************
; Check functions
;***************************************
regLight_1
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l1, 1
    btfsc       LED, 2  ; skip if the 2nd bit not set, set
l* register otherwise
    bsf         l1, 2
    btfsc       LED, 3  ; skip if the 2nd bit not set, set
l* register otherwise
    bsf         l1, 3
    return

regLight_2
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
```

```
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l2, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l2, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l2, 3
    return

regLight_3
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l3, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l3, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l3, 3
    return

regLight_4
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l4, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l4, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l4, 3
    return

regLight_5
    ; check the led register and log the info onto the l*
register
```

```
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l5, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l5, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l5, 3
    return

regLight_6
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l6, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l6, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l6, 3
    return

regLight_7
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l7, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l7, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l7, 3
    return

regLight_8
```

```
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l8, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l8, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l8, 3
    return

regLight_9
    ; check the led register and log the info onto the l*
register
    ; the l* register already cleared if the light is
present
    btfsc       LED, 1  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l9, 1
    btfsc       LED, 2  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l9, 2
    btfsc       LED, 3  ; skip if the 1st bit not set, set
l* register otherwise
    bsf         l9, 3
    return

;**************************************
; Result checking
;**************************************
checkLog
        call    RESULTS
        delayForTimeInterval 6
        goto    DisplayOptions

logLight_1
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc       l1, 0
    movlw       "N"
    btfsc       l1, 1
    movlw       "L"
```

```
    btfsc       l1, 2
    movlw       "F"
    btfsc       l1, 3
    movlw       "P"

    call WR_DATA
    return
logLight_2
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc         l2, 0
    movlw       "N"
    btfsc         l2, 1
    movlw       "L"
    btfsc         l2, 2
    movlw       "F"
    btfsc         l2, 3
    movlw       "P"
    call WR_DATA
   return

logLight_3
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc         l3, 0
    movlw       "N"
    btfsc         l3, 1
    movlw       "L"
    btfsc         l3, 2
    movlw       "F"
    btfsc         l3, 3
    movlw       "P"
    call WR_DATA
   return

logLight_4
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc         l4, 0
    movlw       "N"
    btfsc         l4, 1
    movlw       "L"
    btfsc         l4, 2
    movlw       "F"
    btfsc         l4, 3
    movlw       "P"
```

```
    call WR_DATA
   return

logLight_5
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc          l5, 0
    movlw        "N"
    btfsc          l5, 1
    movlw        "L"
    btfsc          l5, 2
    movlw        "F"
    btfsc          l5, 3
    movlw        "P"
    call WR_DATA
   return

logLight_6
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc          l6, 0
    movlw        "N"
    btfsc          l6, 1
    movlw        "L"
    btfsc          l6, 2
    movlw        "F"
    btfsc          l6, 3
    movlw        "P"
    call WR_DATA
   return

logLight_7
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc          l7, 0
    movlw        "N"
    btfsc          l7, 1
    movlw        "L"
    btfsc          l7, 2
    movlw        "F"
    btfsc          l7, 3
    movlw        "P"
    call WR_DATA
   return

logLight_8
```

```
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc         l8, 0
    movlw       "N"
    btfsc         l8, 1
    movlw       "L"
    btfsc         l8, 2
    movlw       "F"
    btfsc         l8, 3
    movlw       "P"
    call WR_DATA
   return

logLight_9
    ; provide the logging funcion of reading and parsing
each l* register
    btfsc         l9, 0
    movlw       "N"
    btfsc         l9, 1
    movlw       "L"
    btfsc         l9, 2
    movlw       "F"
    btfsc        l9, 3
    movlw       "P"
    call WR_DATA
   return

logLights
    call        logLight_1
    call        logLight_2
    call        logLight_3
    call        logLight_4
    call        logLight_5
    call        logLight_6
    call        logLight_7
    call        logLight_8
    call        logLight_9
    return

;****************************************
; REEPROM module
;****************************************
writeEEPROM
    ; you have to set the EADDR register yourslef in the
code
    ; you have to set the EDATA by yourself
```

```
    banksel     EECON1
    bsf         STATUS, RP1
    bsf         STATUS, RP0     ; bank3
    btfsc       EECON1, WR      ; wait for write to finish
    goto        $-1

    banksel     EEADR
    bcf         STATUS, RP1     ; bank2
    movf        EADDR, W
    movwf       EEADR
    movf        EDATA, W
    movwf       EEDATA

    bsf         STATUS, RP1     ; bank3
    bcf         EECON1, EEPGD
    bsf         EECON1, WREN

    bcf         INTCON, GIE

    movlw       0x55
    movwf       EECON2
    movlw       0xAA
    movwf       EECON2

    bsf         EECON1, WR
    bsf         INTCON, GIE

    btfsc       EECON1, WR      ; wait for write to finish
    goto        $-1

    banksel     PORTA
    return

readEEPROM
    ; set the EADDR to read, output data to W
    banksel     EEDATA
    movf        EADDR, W
    movwf       EEADR

    banksel     EECON1
    bsf         EECON1, RD

    banksel     EEDATA
    movf        EEDATA, W
```

```
    banksel     PORTA
    return

loadEEPROM_1
    ; set the run number
    movlw       b'00000001'
    movwf       EDATA
    movlw       0x00
    movwf       EADDR
    call        writeEEPROM

    ; set the light parameters
    movf        l1, W
    movwf       EDATA
    movlw       0x01
    movwf       EADDR
    call        writeEEPROM

    movf        l2, W
    movwf       EDATA
    movlw       0x02
    movwf       EADDR
    call        writeEEPROM

    movf        l3, W
    movwf       EDATA
    movlw       0x03
    movwf       EADDR
    call        writeEEPROM

    movf        l4, W
    movwf       EDATA
    movlw       0x04
    movwf       EADDR
    call        writeEEPROM

    movf        l5, W
    movwf       EDATA
    movlw       0x05
    movwf       EADDR
    call        writeEEPROM

    movf        l6, W
    movwf       EDATA
    movlw       0x06
    movwf       EADDR
```

```
    call        writeEEPROM

    movf        l7, W
    movwf       EDATA
    movlw       0x07
    movwf       EADDR
    call        writeEEPROM

    movf        l8, W
    movwf       EDATA
    movlw       0x08
    movwf       EADDR
    call        writeEEPROM

    movf        l9, W
    movwf       EDATA
    movlw       0x09
    movwf       EADDR
    call        writeEEPROM

    ; set the time used
    movf        totalSecond, W
    movwf       EDATA
    movlw       0x0A
    movwf       EADDR
    call        writeEEPROM

    ; set the number of lights
    movf        LED_Counter, W
    movwf       EDATA
    movlw       0x0B
    movwf       EADDR
    call        writeEEPROM
return

readEEPROM_1
    ; read the registeres, and load into corresponding reg

    ; read turn
    movlw       0x00
    movwf       EADDR
    call        readEEPROM
    movwf       turn

    ; read lights
    movlw       0x01
```

```
movwf       EADDR
call        readEEPROM
movwf       l1

movlw       0x02
movwf       EADDR
call        readEEPROM
movwf       l2

movlw       0x03
movwf       EADDR
call        readEEPROM
movwf       l3

movlw       0x04
movwf       EADDR
call        readEEPROM
movwf       l4

movlw       0x05
movwf       EADDR
call        readEEPROM
movwf       l5

movlw       0x06
movwf       EADDR
call        readEEPROM
movwf       l6

movlw       0x07
movwf       EADDR
call        readEEPROM
movwf       l7

movlw       0x08
movwf       EADDR
call        readEEPROM
movwf       l8

movlw       0x09
movwf       EADDR
call        readEEPROM
movwf       l9

movlw       0x0A
movwf       EADDR
```

```
call        readEEPROM
movwf       totalSecond


movlw       0x0B
movwf       EADDR
call        readEEPROM
movwf       LED_Counter

return

END
```

TEXAS INSTRUMENTS

CD4067B, CD4097B Types

Data sheet acquired from Harris Semiconductor
SCHS052B – Revised June 2003

CMOS Analog Multiplexers/Demultiplexers

High-Voltage Types (20-Volt Rating)

CD4067B – Single 16-Channel Multiplexer/Demultiplexer
CD4097B – Differential 8-Channel Multiplexer/Demultiplexer

■ CD4067B and CD4097B CMOS analog multiplexers/demultiplexers* are digitally controlled analog switches having low ON impedance, low OFF leakage current, and internal address decoding. In addition, the ON resistance is relatively constant over the full input-signal range.

The CD4067B is a 16-channel multiplexer with four binary control inputs, A,B,C,D, and an inhibit input, arranged so that any combination of the inputs selects one switch.

The CD4097B is a differential 8-channel multiplexer having three binary control inputs A, B, C, and an inhibit input. The inputs permit selection of one of eight pairs of switches.

A logic "1" present at the inhibit input turns all channels off.

The CD4067B and CD4097B types are supplied in 24-lead hermetic dual-in-line ceramic packages (F3A suffix), 24-lead dual-in-line plastic packages (E suffix), 24-lead small-outline packages (M, M96, and NSR suffixes), and 24-lead thin shrink small-outline packages (P and PWR suffixes).

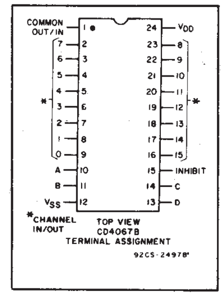
*When these devices are used as demultiplexers, the channel in/out terminals are the outputs and the common out/in terminals are the inputs.

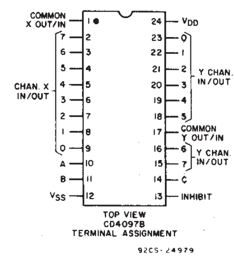
Features:

- Low ON resistance: 125 Ω (typ.) over 15 $V_{p\text{-}p}$ signal-input range for $V_{DD}-V_{SS}$=15 V
- High OFF resistance: channel leakage of ±10 pA (typ.) @ $V_{DD}-V_{SS}$=10 V
- Matched switch characteristics: R_{ON}=5 Ω (typ.) for $V_{DD}-V_{SS}$=15 V
- Very low quiescent power dissipation under all digital-control input and supply conditions: 0.2 μW (typ.) @ $V_{DD}-V_{SS}$=10 V
- Binary address decoding on chip
- 5-V, 10-V, and 15-V parametric ratings
- 100% tested for quiescent current at 20 V
- Standardized symmetrical output characteristics
- Maximum input current of 1 μA at 18 V over full package temperature range; 100 nA at 18 V and 25°C
- Meets all requirements of JEDEC Tentative Standard No. 13B, "Standard Specifications for Description of 'B' Series CMOS Devices"

Applications:

- Analog and digital multiplexing and demultiplexing
- A/D and D/A conversion
- Signal gating





Recommended Operating Conditions at T_A = 25°C (Unless Otherwise Specified)

For maximum reliability, nominal operating conditions should be selected so that operation is always within the following ranges. Values shown apply to all types except as noted.

Characteristic	Min.	Max.	Units
Supply-Voltage Range (T_A=Full Package-Temp. Range)	3	18	V
Multiplexer Switch Input Current Capability	–	25	mA
Output Load Resistance	100	–	Ω

NOTE:

In certain applications, the external load-resistor current may include both V_{DD} and signal-line components. To avoid drawing V_{DD} current when switch current flows into the transmission gate inputs, the voltage drop across the bidirectional switch must not exceed 0.8 volt (calculated from R_{ON} values shown in ELECTRICAL CHARACTERISTICS CHART). No V_{DD} current will flow through R_L if the switch current flows into terminal 1 on the CD4067; terminals 1 and 17 on the CD4097.

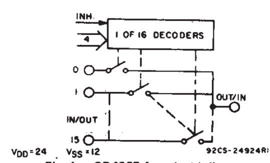


Fig. 1 – CD4067 functional diagram.

CD4067 TRUTH TABLE

A	B	C	D	Inh	Selected Channel
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

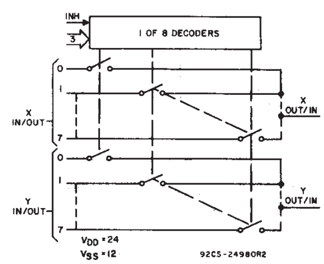


Fig. 2–CD4097 functional diagram.

CD4097 TRUTH TABLE

A	B	C	Inh	Selected Channel
X	X	X	1	None
0	0	0	0	0X, 0Y
1	0	0	0	1X, 1Y
0	1	0	0	2X, 2Y
1	1	0	0	3X, 3Y
0	0	1	0	4X, 4Y
1	0	1	0	5X, 5Y
0	1	1	0	6X, 6Y
1	1	1	0	7X, 7Y

General Purpose Silicon Rectifiers

SMD Diodes Specialist

1N4001-G Thru. 1N4007-G

Voltage: 50 to 1000 V
Current: 1.0 A
RoHS Device

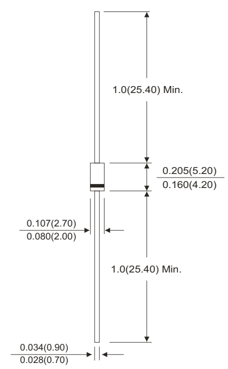
Features

- -Low cost construction.
- -Fast forward voltage drop.
- -Low reverse leakage.
- -High forward surge current capability.
- -High soldering temperature guarantee: 260 °C/10 seconds, 0.375”(9.5mm) lead length at 5lbs(2.3kg) tension.

Mechanical data

- -Case: transfer molded plastic, DO-41
- -Epoxy: UL 94V-0 rate flame retardant
- -Polarity: Indicated by cathode band
- -Lead: Plated axial lead, solderable per MIL-STD-202E, method 208C
- -Mounting position: Any
- -Weight: 0.012ounce, 0.33 grams

DO-41



Dimensions in inches and (millimeter)

Electrical Characteristics (at T_A=25 °C unless otherwise noted)

Ratings at 25 °C ambient temperature unless otherwise specified.
Single phase, half wave, 60Hz, resistive or inductive load.
For capacitive load derate current by 20%.

Parameter	Symbol	1N4001-G	1N4002-G	1N4003-G	1N4004-G	1N4005-G	1N4006-G	1N4007-G	Unit
Maximum Repetitive Peak Reverse Voltage	V_{RRM}	50	100	200	400	600	800	1000	V
Maximum RMS Voltage	V_{RMS}	35	70	140	280	420	560	700	V
Maximum DC Blocking Voltage	V_{DC}	50	100	200	400	600	800	1000	V
Maximum Average Forward Rectified Current 0.375"(9.5mm) Lead Length @T_A=55 °C	$I_{(AV)}$	1.0							A
Peak Forward Surge Current, 8.3mS single half sine-wave superimposed on rated load (JEDEC method)	I_{FSM}	30							A
Maximum Instantaneous Forward Voltage @1.0A	V_F	1.1							V
Maximum DC Reverse Current at Rated DC Blocking voltage per element, T_A=25 °C	I_R	5.0							μA
Maximum DC Reverse Current at Rated DC Blocking voltage per element, T_A=100 °C	I_R	50							μA
Maximum Full Load Reverse Current,full cycle average 0.375"(9.5mm)lead length at T_L=75 °C	$I_{R(AV)}$	30							μA
Typical Junction Capacitance (Note 1)	C_J	15							pF
Typical Thermal Resistance (Note 2)	$R_{\theta JA}$	60							°C/W
Operating Temperature Range	T_J	-55 ~ +150							°C
Storage Ttemperature Range	T_{STG}	-55 ~ +150							°C

NOTES:
1. Measured at 1.0MHz and Applied Reverse Voltage of 4.0V DC.
2. Thermal Resistance from junction to terminal 6.0mm^2 copper pads to each terminal.

REV:A

General Purpose Silicon Rectifiers

COMCHIP
SMD Diodes Specialist

Rating and Characteristic Curves (1N4001-G Thru. 1N4007-G)

Fig.1 Typical Forward Current Derating Curve

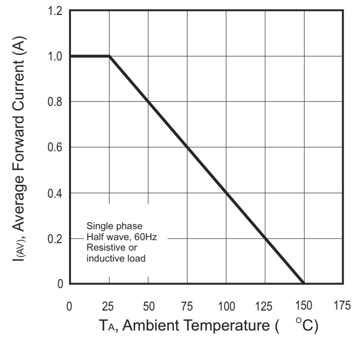


Fig.2 Maximum. Non-Repetitive Peak Forward Surge Current

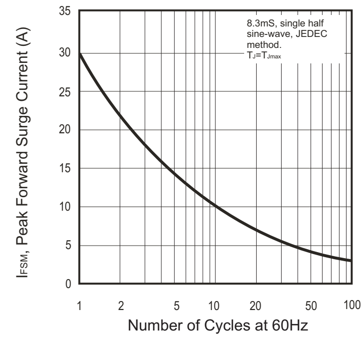


Fig.3 Typical Instantaneous Forward Characteristics

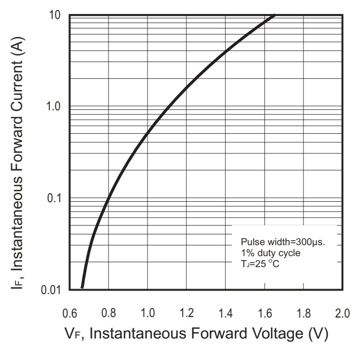


Fig.4 Typical Reverse Characteristics

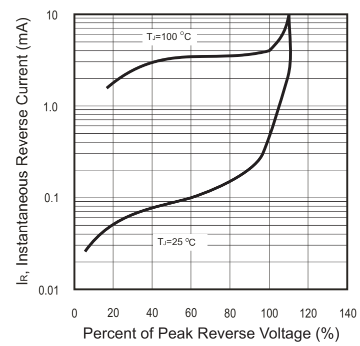
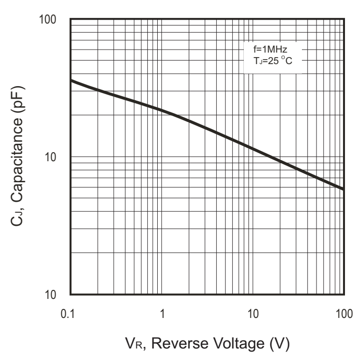


Fig.5 Typical Junction Capacitance

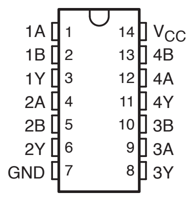


SN54HC00, SN74HC00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

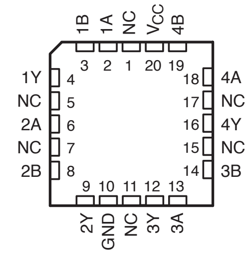
SCLS181E – DECEMBER 1982 – REVISED AUGUST 2003

- **Wide Operating Voltage Range of 2 V to 6 V**
- **Outputs Can Drive Up To 10 LSTTL Loads**
- **Low Power Consumption, 20-µA Max I_{CC}**
- **Typical t_{pd} = 8 ns**
- **±4-mA Output Drive at 5 V**
- **Low Input Current of 1 µA Max**

SN54HC00 . . . J OR W PACKAGE
SN74HC00 . . . D, DB, N, NS, OR PW PACKAGE
(TOP VIEW)



SN54HC00 . . . FK PACKAGE
(TOP VIEW)



NC – No internal connection

description/ordering information

The ’HC00 devices contain four independent 2-input NAND gates. They perform the Boolean function $Y = \overline{A \bullet B}$ or $Y = \overline{A} + \overline{B}$ in positive logic.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
–40°C to 85°C	PDIP – N	Tube of 25	SN74HC00N	SN74HC00N
	SOIC – D	Tube of 50	SN74HC00D	HC00
		Reel of 2500	SN74HC00DR	
		Reel of 250	SN74HC00DT	
	SOP – NS	Reel of 2000	SN74HC00NSR	HC00
	SSOP – DB	Reel of 2000	SN74HC00DBR	HC00
	TSSOP – PW	Tube of 90	SN74HC00PW	HC00
		Reel of 2000	SN74HC00PWR	
		Reel of 250	SN74HC00PWT	
–55°C to 125°C	CDIP – J	Tube of 25	SNJ54HC00J	SNJ54HC00J
	CFP – W	Tube of 150	SNJ54HC00W	SNJ54HC00W
	LCCC – FK	Tube of 55	SNJ54HC00FK	SNJ54HC00FK

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

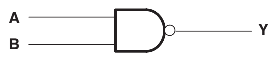
SN54HC00, SN74HC00
QUADRUPLE 2-INPUT POSITIVE-NAND GATES

SCLS181E – DECEMBER 1982 – REVISED AUGUST 2003

FUNCTION TABLE
(each gate)

INPUTS		OUTPUT
A	B	Y
H	H	L
L	X	H
X	L	H

logic diagram (positive logic)



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)†

Supply voltage range, V_{CC} . . . . –0.5 V to 7 V
Input clamp current, I_{IK} ($V_I < 0$ or $V_I > V_{CC}$) (see Note 1) . . . . ±20 mA
Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{CC}$) (see Note 1) . . . . ±20 mA
Continuous output current, I_O ($V_O = 0$ to V_{CC}) . . . . ±25 mA
Continuous current through V_{CC} or GND . . . . ±50 mA
Package thermal impedance, θ_{JA} (see Note 2): D package . . . . 86°C/W
DB package . . . . 96°C/W
N package . . . . 80°C/W
NS package . . . . 76°C/W
PW package . . . . 113°C/W
Storage temperature range, T_{stg} . . . . –65°C to 150°C

† Stresses beyond those listed under “absolute maximum ratings” may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under “recommended operating conditions” is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

NOTES: 1. The input and output voltage ratings may be exceeded if the input and output current ratings are observed.
2. The package thermal impedance is calculated in accordance with JESD 51-7.

recommended operating conditions (see Note 3)

			SN54HC00			SN74HC00			UNIT
			MIN	NOM	MAX	MIN	NOM	MAX	
V_{CC}	Supply voltage		2	5	6	2	5	6	V
V_{IH}	High-level input voltage	V_{CC} = 2 V	1.5			1.5			V
		V_{CC} = 4.5 V	3.15			3.15			
		V_{CC} = 6 V	4.2			4.2			
V_{IL}	Low-level input voltage	V_{CC} = 2 V			0.5			0.5	V
		V_{CC} = 4.5 V			1.35			1.35	
		V_{CC} = 6 V			1.8			1.8	
V_I	Input voltage		0		V_{CC}	0		V_{CC}	V
V_O	Output voltage		0		V_{CC}	0		V_{CC}	V
$\Delta t/\Delta v$	Input transition rise/fall time	V_{CC} = 2 V			1000			1000	ns
		V_{CC} = 4.5 V			500			500	
		V_{CC} = 6 V			400			400	
T_A	Operating free-air temperature		–55		125	–40		85	°C

NOTE 3: All unused inputs of the device must be held at V_{CC} or GND to ensure proper device operation. Refer to the TI application report, *Implications of Slow or Floating CMOS Inputs*, literature number SCBA004.

LM158/LM258/LM358/LM2904 Low Power Dual Operational Amplifiers

Check for Samples: LM158-N, LM258-N, LM2904-N, LM358-N

FEATURES

- **Available in 8-Bump DSBGA Chip-Sized Package, (See AN-1112 (SNVA009))**
- **Internally Frequency Compensated for Unity Gain**
- **Large DC Voltage Gain: 100 dB**
- **Wide Bandwidth (Unity Gain): 1 MHz (Temperature Compensated)**
- **Wide Power Supply Range:**
  - **Single Supply: 3V to 32V**
  - **Or Dual Supplies: ±1.5V to ±16V**
- **Very Low Supply Current Drain (500 μA)—Essentially Independent of Supply Voltage**
- **Low Input Offset Voltage: 2 mV**
- **Input Common-Mode Voltage Range Includes Ground**
- **Differential Input Voltage Range Equal to the Power Supply Voltage**
- **Large Output Voltage Swing**

UNIQUE CHARACTERISTICS

- **In the Llinear Mode the Input Common-Mode Voltage Range Includes Ground and the Output Voltage Can Also Swing to Ground, even though Operated from Only a Single Power Supply Voltage.**
- **The Unity Gain Cross Frequency is Temperature Compensated.**
- **The Input Bias Current is also Temperature Compensated.**

ADVANTAGES

- **Two Internally Compensated Op Amps**
- **Eliminates Need for Dual Supplies**
- **Allows Direct Sensing Near GND and V_{OUT} Also Goes to GND**
- **Compatible with All Forms of Logic**
- **Power Drain Suitable for Battery Operation**

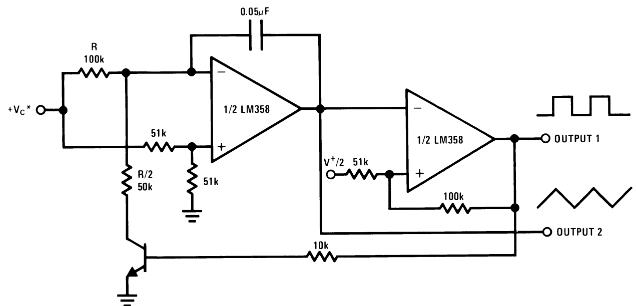
DESCRIPTION

The LM158 series consists of two independent, high gain, internally frequency compensated operational amplifiers which were designed specifically to operate from a single power supply over a wide range of voltages. Operation from split power supplies is also possible and the low power supply current drain is independent of the magnitude of the power supply voltage.

Application areas include transducer amplifiers, dc gain blocks and all the conventional op amp circuits which now can be more easily implemented in single power supply systems. For example, the LM158 series can be directly operated off of the standard +5V power supply voltage which is used in digital systems and will easily provide the required interface electronics without requiring the additional ±15V power supplies.

The LM358 and LM2904 are available in a chip sized package (8-Bump DSBGA) using TI's DSBGA package technology.

Voltage Controlled Oscillator (VCO)



These devices have limited built-in ESD protection. The leads should be shorted together or the device placed in conductive foam during storage or handling to prevent electrostatic damage to the MOS gates.

3.1.5. Catastrophic Failure Protection

Should a component failure occur, the power supply should not exhibit any of the following:

- Flame
- Excessive smoke
- Charred PCB
- Fused PCB conductor
- Startling noise
- Emission of molten material

3.2. DC Output

3.2.1. DC Voltage Regulation

The DC output voltages shall remain within the regulation ranges shown in Table 2 when measured at the load end of the output connectors under all line, load, and environmental conditions. The voltage regulation limits shall be maintained under continuous operation for any steady state temperature and operating conditions specified in Section 5.

Table 2. DC Output Voltage Regulation

Output	Range	Min.	Nom.	Max.	Unit
+12V1DC	±5%	+11.40	+12.00	+12.60	Volts
+12V2DC (1)	±5%	+11.40	+12.00	+12.60	Volts
+5VDC	±5%	+4.75	+5.00	+5.25	Volts
+3.3VDC (2)	±5%	+3.14	+3.30	+3.47	Volts
-12VDC	±10%	-10.80	-12.00	-13.20	Volts
+5VSB	±5%	+4.75	+5.00	+5.25	Volts

(1) At +12 VDC peak loading, regulation at the +12 VDC output can go to ± 10%.

(2) Voltage tolerance is required at main connector and S-ATA connector (if used).

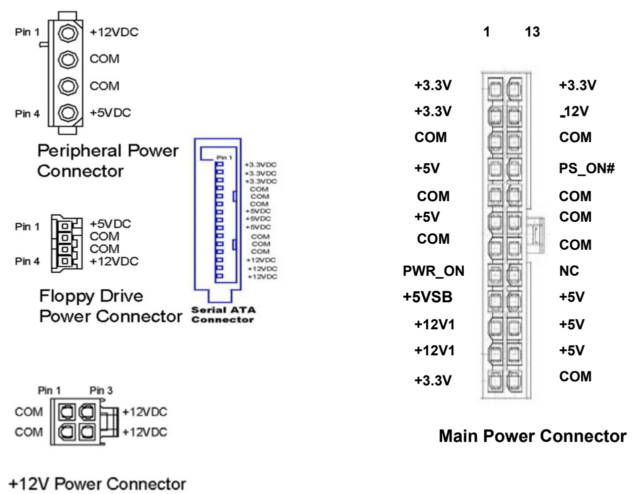
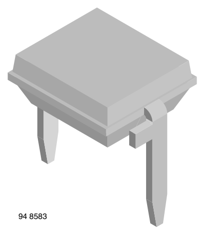


Figure 10. ATX12V Power Supply Connectors
(Pin-side view, not to scale)

Silicon PIN Photodiode



FEATURES

- Package type: leaded
- Package form: top view
- Dimensions (L x W x H in mm): 5.4 x 4.3 x 3.2
- Radiant sensitive area (in mm^2): 7.5
- High photo sensitivity
- High radiant sensitivity
- Suitable for visible and near infrared radiation
- Fast response times
- Angle of half sensitivity: $\varphi = \pm 65°$
- Compliant to RoHS Directive 2002/95/EC and in accordance to WEEE 2002/96/EC

Note

** Please see document "Vishay Material Category Policy": www.vishay.com/doc?99902

DESCRIPTION

BPW34 is a PIN photodiode with high speed and high radiant sensitivity in miniature, flat, top view, clear plastic package. It is sensitive to visible and near infrared radiation. BPW34S is packed in tubes, specifications like BPW34.

APPLICATIONS

- High speed photo detector

PRODUCT SUMMARY			
COMPONENT	**I_{ra} (μA)**	**φ (deg)**	**$\lambda_{0.1}$ (nm)**
BPW34	50	± 65	430 to 1100
BPW34S	50	± 65	430 to 1100

Note

- Test condition see table "Basic Characteristics"

ORDERING INFORMATION			
ORDERING CODE	**PACKAGING**	**REMARKS**	**PACKAGE FORM**
BPW34	Bulk	MOQ: 3000 pcs, 3000 pcs/bulk	Top view
BPW34S	Tube	MOQ: 1800 pcs, 45 pcs/tube	Top view

Note

- MOQ: minimum order quantity

ABSOLUTE MAXIMUM RATINGS (T_{amb} = 25 °C, unless otherwise specified)				
PARAMETER	**TEST CONDITION**	**SYMBOL**	**VALUE**	**UNIT**
Reverse voltage		V_R	60	V
Power dissipation	$T_{amb} \leq 25$ °C	P_V	215	mW
Junction temperature		T_j	100	°C
Operating temperature range		T_{amb}	- 40 to + 100	°C
Storage temperature range		T_{stg}	- 40 to + 100	°C
Soldering temperature	$t \leq 3$ s	T_{sd}	260	°C
Thermal resistance junction/ambient	Connected with Cu wire, 0.14 mm^2	R_{thJA}	350	K/W

BASIC CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)						
PARAMETER	TEST CONDITION	SYMBOL	MIN.	TYP.	MAX.	UNIT
Breakdown voltage	I_R = 100 μA, E = 0	$V_{(BR)}$	60			V
Reverse dark current	V_R = 10 V, E = 0	I_{ro}		2	30	nA
Diode capacitance	V_R = 0 V, f = 1 MHz, E = 0	C_D		70		pF
	V_R = 3 V, f = 1 MHz, E = 0	C_D		25	40	pF
Open circuit voltage	E_e = 1 mW/cm², λ = 950 nm	V_o		350		mV
Temperature coefficient of V_o	E_e = 1 mW/cm², λ = 950 nm	TK_{Vo}		- 2.6		mV/K
Short circuit current	E_A = 1 klx	I_k		70		μA
	E_e = 1 mW/cm², λ = 950 nm	I_k		47		μA
Temperature coefficient of I_k	E_e = 1 mW/cm², λ = 950 nm	TK_{Ik}		0.1		%/K
Reverse light current	E_A = 1 klx, V_R = 5 V	I_{ra}		75		μA
	E_e = 1 mW/cm², λ = 950 nm, V_R = 5 V	I_{ra}	40	50		μA
Angle of half sensitivity		φ		± 65		deg
Wavelength of peak sensitivity		λ_p		900		nm
Range of spectral bandwidth		$\lambda_{0.1}$		430 to 1100		nm
Noise equivalent power	V_R = 10 V, λ = 950 nm	NEP		4 x 10^{-14}		W/√Hz
Rise time	V_R = 10 V, R_L = 1 kΩ, λ = 820 nm	t_r		100		ns
Fall time	V_R = 10 V, R_L = 1 kΩ, λ = 820 nm	t_f		100		ns

BASIC CHARACTERISTICS (T_{amb} = 25 °C, unless otherwise specified)

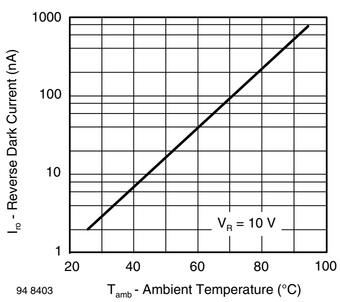


Fig. 1 - Reverse Dark Current vs. Ambient Temperature

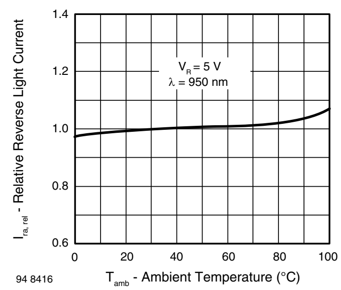


Fig. 2 - Relative Reverse Light Current vs. Ambient Temperature

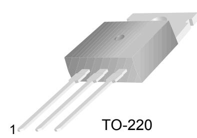
FAIRCHILD SEMICONDUCTOR®

TIP140T/141T/142T

TIP140T/141T/142T

Monolithic Construction With Built In Base-Emitter Shunt Resistors

- High DC Current Gain : h_{FE} = 1000 @ V_{CE} = 4V, I_C = 5A (Min.)
- Industrial Use
- Complement to TIP145T/146T/147T

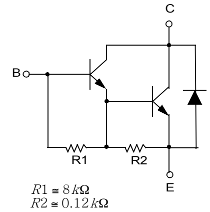


1 TO-220

1.Base 2.Collector 3.Emitter

NPN Epitaxial Silicon Darlington Transistor

Equivalent Circuit



Absolute Maximum Ratings T_C=25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{CBO}	Collector-Base Voltage : TIP140T	60	V
	: TIP141T	80	V
	: TIP142T	100	V
V_{CEO}	Collector-Emitter Voltage : TIP140T	60	V
	: TIP141T	80	V
	: TIP142T	100	V
V_{EBO}	Emitter-Base Voltage	5	V
I_C	Collector Current (DC)	10	A
I_{CP}	Collector Current (Pulse)	15	A
I_B	Base Current (DC)	0.5	A
P_C	Collector Dissipation (T_C=25°C)	80	W
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	- 65 ~ 150	°C

Electrical Characteristics T_C=25°C unless otherwise noted

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
V_{CEO}(sus)	Collector-Emitter Sustaining Voltage					
	: TIP140T	I_C = 30mA, I_B = 0	60			V
	: TIP141T		80			V
	: TIP142T		100			V
I_{CEO}	Collector Cut-off Current					
	: TIP140T	V_{CE} = 30V, I_B = 0			2	mA
	: TIP141T	V_{CE} = 40V, I_B = 0			2	mA
	: TIP142T	V_{CE} = 50V, I_B = 0			2	mA
I_{CBO}	Collector Cut-off Current					
	: TIP140T	V_{CB} = 60V, I_E = 0			1	mA
	: TIP141T	V_{CB} = 80V, I_E = 0			1	mA
	: TIP142T	V_{CB} = 100V, I_E = 0			1	mA
I_{EBO}	Emitter Cut-off Current	V_{BE} = 5V, I_C = 0			2	mA
h_{FE}	DC Current Gain	V_{CE} = 4V, I_C = 5A	1000			mA
		V_{CE} =4V, I_C = 10A	500			
V_{CE}(sat)	Collector-Emitter Saturation Voltage	I_C = 5A, I_B = 10mA			2	V
		I_C = 10A, I_B = 40mA			3	V
V_{BE}(sat)	Base-Emitter Saturation Voltage	I_C = 10A, I_B = 40mA			3.5	V
V_{BE}(on)	Base-Emitter On Voltage	V_{CE} = 4V, I_C = 10A			3	V
t_D	Delay Time	V_{CC} = 30V, I_C = 5A		0.15		µs
t_R	Rise Time	I_{B1} = 20mA		0.55		µs
t_{STG}	Storage Time	I_{B2} = -20mA		2.5		µs
t_F	Fall Time	R_L = 6Ω		2.5		µs

 Rev. B1, December 2002

TIP140T/141T/142T

Typical Characteristics

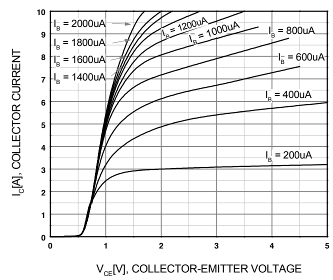


Figure 1. Static Characteristic

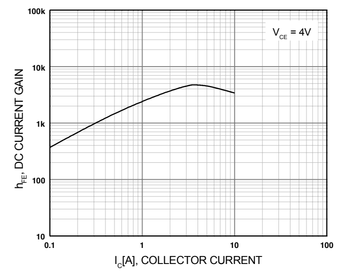


Figure 2. DC current Gain

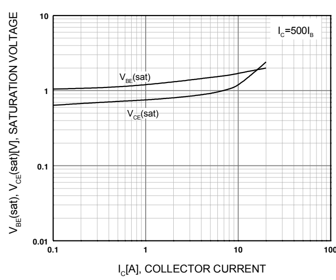


Figure 3. Collector-Emitter Saturation Voltage Base-Emitter Saturation Voltage

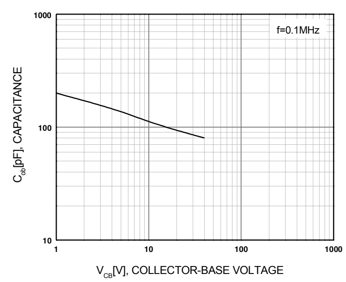


Figure 4. Collector Output Capacitance

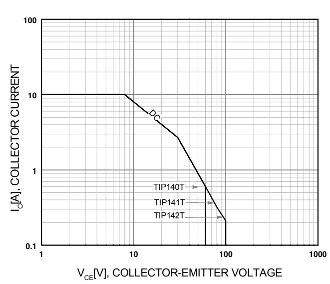


Figure 5. Safe Operating Area

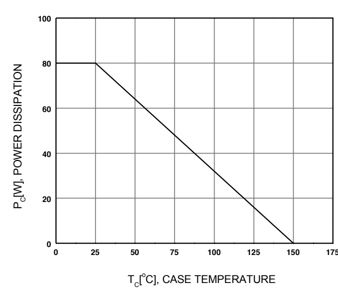


Figure 6. Power Derating

 Rev. B1, December 2002

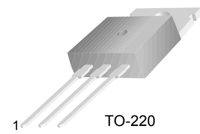
FAIRCHILD
SEMICONDUCTOR®

TIP145T/146T/147T

TIP145T/146T/147T

Monolithic Construction With Built In Base-Emitter Shunt Resistors

- High DC Current Gain : h_{FE} = 1000@ V_{CE} = - 4V, I_C = - 5A (Min.)
- Industrial Use
- Complement to TIP140T/141T/142T

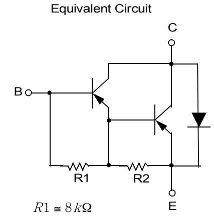


TO-220

1.Base 2.Collector 3.Emitter

PNP Epitaxial Silicon Darlington Transistor

Equivalent Circuit



$R1 \cong 8k\Omega$
$R2 \cong 0.12k\Omega$

Absolute Maximum Ratings T_C=25°C unless otherwise noted

Symbol	Parameter	Value	Units
V_{CBO}	Collector-Base Voltage : TIP145T : TIP146T : TIP147T	- 60 - 80 - 100	V V V
V_{CEO}	Collector-Emitter Voltage : TIP145T : TIP146T : TIP147T	- 60 - 80 - 100	V V V
V_{EBO}	Emitter-Base Voltage	- 5	V
I_C	Collector Current (DC)	- 10	A
I_{CP}	Collector Current (Pulse)	- 15	A
I_B	Base Current (DC)	- 0.5	A
P_C	Collector Dissipation (T_C=25°C)	80	W
T_J	Junction Temperature	150	°C
T_{STG}	Storage Temperature	- 65 ~ 150	°C

Electrical Characteristics T_C=25°C unless otherwise noted

Symbol	Parameter	Test Condition	Min.	Typ.	Max.	Units
V_{CEO}(sus)	Collector-Emitter Sustaining Voltage : TIP145T : TIP146T : TIP147T	I_C = - 30mA, I_B = 0	 - 60 - 80 - 100			 V V V
I_{CEO}	Collector Cut-off Current : TIP145T : TIP146T : TIP147T	 V_{CE} = - 30V, I_B = 0 V_{CE} = - 40V, I_B = 0 V_{CE} = - 50V, I_B = 0			 - 2 - 2 - 2	 mA mA mA
I_{CBO}	Collector Cut-off Current : TIP145T : TIP146T : TIP147T	 V_{CB} = - 60V, I_E = 0 V_{CB} = - 80V, I_E = 0 V_{CB} = - 100V, I_E = 0			 - 1 - 1 - 1	 mA mA mA
I_{EBO}	Emitter Cut-off Current	V_{BE} = - 5V, I_C = 0			- 2	mA
h_{FE}	DC Current Gain	V_{CE} = - 4V, I_C = - 5A V_{CE} = - 4V, I_C = - 10A	1000 500			
V_{CE}(sat)	Collector-Emitter Saturation Voltage	I_C = - 5A, I_B = - 10mA I_C = - 10A, I_B = - 40mA			- 2 - 3	V V
V_{BE}(sat)	Base-Emitter Saturation Voltage	I_C = - 10A, I_B = - 40mA			- 3.5	V
V_{BE}(on)	Base-Emitter On Voltage	V_{CE} = - 4V, I_C = - 10A			- 3	V
t_D	Delay Time	V_{CC} = - 30V, I_C = - 5A I_{B1} = -20mA, I_{B2} = 20mA R_L = 6Ω		0.15		μs
t_R	Rise Time			0.55		μs
t_{STG}	Storage Time			2.5		μs
t_F	Fall Time			2.5		μs

 Rev. B, December 2002

Typical Characteristics

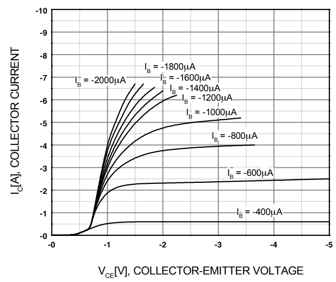


Figure 1. Static Characteristic

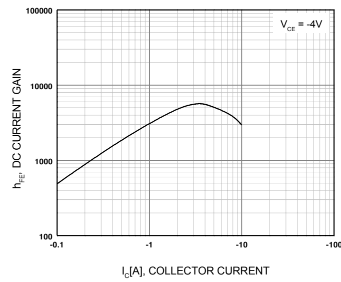


Figure 2. DC current Gain

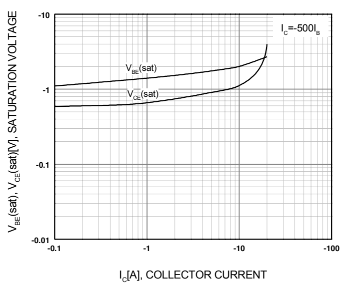


Figure 3. Collector-Emitter Saturation Voltage Base-Emitter Saturation Voltage

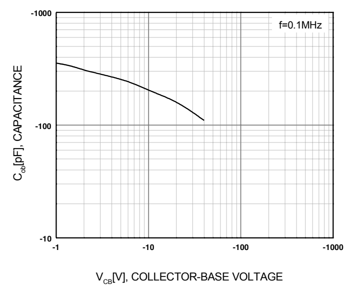


Figure 4. Collector Output Capacitance

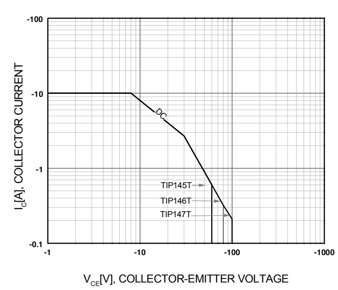


Figure 5. Safe Operating Area

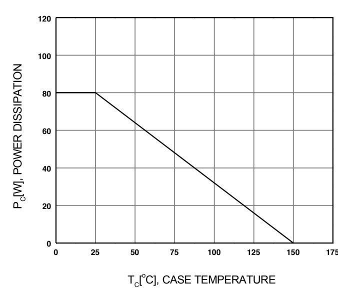


Figure 6. Power Derating

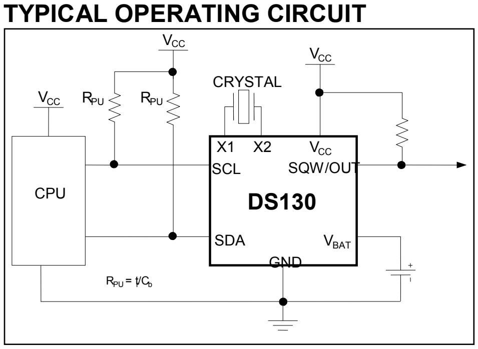
 Rev. B, December 2002

DS1307
64 x 8, Serial, I²C Real-Time Clock

GENERAL DESCRIPTION

The DS1307 serial real-time clock (RTC) is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I²C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

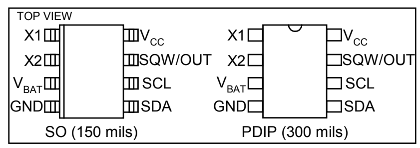
TYPICAL OPERATING CIRCUIT



FEATURES

- Real-Time Clock (RTC) Counts Seconds, Minutes, Hours, Date of the Month, Month, Day of the week, and Year with Leap-Year Compensation Valid Up to 2100
- 56-Byte, Battery-Backed, General-Purpose RAM with Unlimited Writes
- I²C Serial Interface
- Programmable Square-Wave Output Signal
- Automatic Power-Fail Detect and Switch Circuitry
- Consumes Less than 500nA in Battery-Backup Mode with Oscillator Running
- Optional Industrial Temperature Range: -40°C to +85°C
- Available in 8-Pin Plastic DIP or SO
- Underwriters Laboratories (UL) Recognized

PIN CONFIGURATIONS



ORDERING INFORMATION

PART	TEMP RANGE	VOLTAGE (V)	PIN-PACKAGE	TOP MARK*
DS1307+	0°C to +70°C	5.0	8 PDIP (300 mils)	DS1307
DS1307N+	-40°C to +85°C	5.0	8 PDIP (300 mils)	DS1307N
DS1307Z+	0°C to +70°C	5.0	8 SO (150 mils)	DS1307
DS1307ZN+	-40°C to +85°C	5.0	8 SO (150 mils)	DS1307N
DS1307Z+T&R	0°C to +70°C	5.0	8 SO (150 mils) Tape and Reel	DS1307
DS1307ZN+T&R	-40°C to +85°C	5.0	8 SO (150 mils) Tape and Reel	DS1307N

+Denotes a lead-free/RoHS-compliant package.
**A "+" anywhere on the top mark indicates a lead-free package. An "N" anywhere on the top mark indicates an industrial temperature range device.*

For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maximintegrated.com.

REV: 100208

PIC16F87X

28/40-Pin 8-Bit CMOS FLASH Microcontrollers

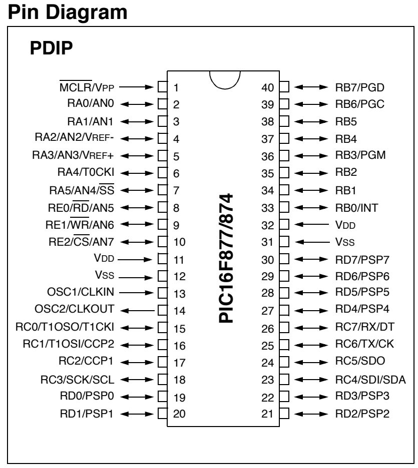
Devices Included in this Data Sheet:

- PIC16F873
- PIC16F874
- PIC16F876
- PIC16F877

Microcontroller Core Features:

- High performance RISC CPU
- Only 35 single word instructions to learn
- All single cycle instructions except for program branches which are two cycle
- Operating speed: DC - 20 MHz clock input
  DC - 200 ns instruction cycle
- Up to 8K x 14 words of FLASH Program Memory,
  Up to 368 x 8 bytes of Data Memory (RAM)
  Up to 256 x 8 bytes of EEPROM Data Memory
- Pinout compatible to the PIC16C73B/74B/76/77
- Interrupt capability (up to 14 sources)
- Eight level deep hardware stack
- Direct, indirect and relative addressing modes
- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Programmable code protection
- Power saving SLEEP mode
- Selectable oscillator options
- Low power, high speed CMOS FLASH/EEPROM technology
- Fully static design
- In-Circuit Serial Programming™ (ICSP) via two pins
- Single 5V In-Circuit Serial Programming capability
- In-Circuit Debugging via two pins
- Processor read/write access to program memory
- Wide operating voltage range: 2.0V to 5.5V
- High Sink/Source Current: 25 mA
- Commercial, Industrial and Extended temperature ranges
- Low-power consumption:
  - < 0.6 mA typical @ 3V, 4 MHz
  - 20 µA typical @ 3V, 32 kHz
  - < 1 µA typical standby current

Pin Diagram



Peripheral Features:

- Timer0: 8-bit timer/counter with 8-bit prescaler
- Timer1: 16-bit timer/counter with prescaler, can be incremented during SLEEP via external crystal/clock
- Timer2: 8-bit timer/counter with 8-bit period register, prescaler and postscaler
- Two Capture, Compare, PWM modules
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- 10-bit multi-channel Analog-to-Digital converter
- Synchronous Serial Port (SSP) with SPI™ (Master mode) and I^{2}C™ (Master/Slave)
- Universal Synchronous Asynchronous Receiver Transmitter (USART/SCI) with 9-bit address detection
- Parallel Slave Port (PSP) 8-bits wide, with external RD, WR and CS controls (40/44-pin only)
- Brown-out detection circuitry for Brown-out Reset (BOR)

PIC16F87X

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS(4)	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs CLKOUT which has 1/4 the frequency of OSC1, and denotes the instruction cycle rate.
$\overline{MCLR}$/VPP	1	2	18	I/P	ST	Master Clear (Reset) input or programming voltage input. This pin is an active low RESET to the device.
						PORTA is a bi-directional I/O port.
RA0/AN0	2	3	19	I/O	TTL	RA0 can also be analog input0.
RA1/AN1	3	4	20	I/O	TTL	RA1 can also be analog input1.
RA2/AN2/VREF-	4	5	21	I/O	TTL	RA2 can also be analog input2 or negative analog reference voltage.
RA3/AN3/VREF+	5	6	22	I/O	TTL	RA3 can also be analog input3 or positive analog reference voltage.
RA4/T0CKI	6	7	23	I/O	ST	RA4 can also be the clock input to the Timer0 timer/counter. Output is open drain type.
RA5/$\overline{SS}$/AN4	7	8	24	I/O	TTL	RA5 can also be analog input4 or the slave select for the synchronous serial port.
						PORTB is a bi-directional I/O port. PORTB can be software programmed for internal weak pull-up on all inputs.
RB0/INT	33	36	8	I/O	TTL/ST(1)	RB0 can also be the external interrupt pin.
RB1	34	37	9	I/O	TTL	
RB2	35	38	10	I/O	TTL	
RB3/PGM	36	39	11	I/O	TTL	RB3 can also be the low voltage programming input.
RB4	37	41	14	I/O	TTL	Interrupt-on-change pin.
RB5	38	42	15	I/O	TTL	Interrupt-on-change pin.
RB6/PGC	39	43	16	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming clock.
RB7/PGD	40	44	17	I/O	TTL/ST(2)	Interrupt-on-change pin or In-Circuit Debugger pin. Serial programming data.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

v

TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION (CONTINUED)

Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
						PORTC is a bi-directional I/O port.
RC0/T1OSO/T1CKI	15	16	32	I/O	ST	RC0 can also be the Timer1 oscillator output or a Timer1 clock input.
RC1/T1OSI/CCP2	16	18	35	I/O	ST	RC1 can also be the Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output.
RC2/CCP1	17	19	36	I/O	ST	RC2 can also be the Capture1 input/Compare1 output/PWM1 output.
RC3/SCK/SCL	18	20	37	I/O	ST	RC3 can also be the synchronous serial clock input/output for both SPI and I^2C modes.
RC4/SDI/SDA	23	25	42	I/O	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	24	26	43	I/O	ST	RC5 can also be the SPI Data Out (SPI mode).
RC6/TX/CK	25	27	44	I/O	ST	RC6 can also be the USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	26	29	1	I/O	ST	RC7 can also be the USART Asynchronous Receive or Synchronous Data.
						PORTD is a bi-directional I/O port or parallel slave port when interfacing to a microprocessor bus.
RD0/PSP0	19	21	38	I/O	ST/TTL(3)	
RD1/PSP1	20	22	39	I/O	ST/TTL(3)	
RD2/PSP2	21	23	40	I/O	ST/TTL(3)	
RD3/PSP3	22	24	41	I/O	ST/TTL(3)	
RD4/PSP4	27	30	2	I/O	ST/TTL(3)	
RD5/PSP5	28	31	3	I/O	ST/TTL(3)	
RD6/PSP6	29	32	4	I/O	ST/TTL(3)	
RD7/PSP7	30	33	5	I/O	ST/TTL(3)	
						PORTE is a bi-directional I/O port.
RE0/$\overline{RD}$/AN5	8	9	25	I/O	ST/TTL(3)	RE0 can also be read control for the parallel slave port, or analog input5.
RE1/$\overline{WR}$/AN6	9	10	26	I/O	ST/TTL(3)	RE1 can also be write control for the parallel slave port, or analog input6.
RE2/$\overline{CS}$/AN7	10	11	27	I/O	ST/TTL(3)	RE2 can also be select control for the parallel slave port, or analog input7.
V_{SS}	12,31	13,34	6,29	P	—	Ground reference for logic and I/O pins.
V_{DD}	11,32	12,35	7,28	P	—	Positive supply for logic and I/O pins.
NC	—	1,17,28, 40	12,13, 33,34		—	These pins are not internally connected. These pins should be left unconnected.

Legend: I = input O = output I/O = input/output P = power
— = Not used TTL = TTL input ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as an external interrupt.
2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.
3: This buffer is a Schmitt Trigger input when configured as general purpose I/O and a TTL input when used in the Parallel Slave Port mode (for interfacing to a microprocessor bus).
4: This buffer is a Schmitt Trigger input when configured in RC oscillator mode and a CMOS input otherwise.

FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP

Bank 0	File Address	Bank 1	File Address	Bank 2	File Address	Bank 3	File Address
Indirect addr.(*)	00h	Indirect addr.(*)	80h	Indirect addr.(*)	100h	Indirect addr.(*)	180h
TMR0	01h	OPTION_REG	81h	TMR0	101h	OPTION_REG	181h
PCL	02h	PCL	82h	PCL	102h	PCL	182h
STATUS	03h	STATUS	83h	STATUS	103h	STATUS	183h
FSR	04h	FSR	84h	FSR	104h	FSR	184h
PORTA	05h	TRISA	85h		105h		185h
PORTB	06h	TRISB	86h	PORTB	106h	TRISB	186h
PORTC	07h	TRISC	87h		107h		187h
PORTD(1)	08h	TRISD(1)	88h		108h		188h
PORTE(1)	09h	TRISE(1)	89h		109h		189h
PCLATH	0Ah	PCLATH	8Ah	PCLATH	10Ah	PCLATH	18Ah
INTCON	0Bh	INTCON	8Bh	INTCON	10Bh	INTCON	18Bh
PIR1	0Ch	PIE1	8Ch	EEDATA	10Ch	EECON1	18Ch
PIR2	0Dh	PIE2	8Dh	EEADR	10Dh	EECON2	18Dh
TMR1L	0Eh	PCON	8Eh	EEDATH	10Eh	Reserved(2)	18Eh
TMR1H	0Fh		8Fh	EEADRH	10Fh	Reserved(2)	18Fh
T1CON	10h		90h	General Purpose Register 16 Bytes	110h	General Purpose Register 16 Bytes	190h
TMR2	11h	SSPCON2	91h		111h		191h
T2CON	12h	PR2	92h		112h		192h
SSPBUF	13h	SSPADD	93h		113h		193h
SSPCON	14h	SSPSTAT	94h		114h		194h
CCPR1L	15h		95h		115h		195h
CCPR1H	16h		96h		116h		196h
CCP1CON	17h		97h		117h		197h
RCSTA	18h	TXSTA	98h		118h		198h
TXREG	19h	SPBRG	99h		119h		199h
RCREG	1Ah		9Ah		11Ah		19Ah
CCPR2L	1Bh		9Bh		11Bh		19Bh
CCPR2H	1Ch		9Ch		11Ch		19Ch
CCP2CON	1Dh		9Dh		11Dh		19Dh
ADRESH	1Eh	ADRESL	9Eh		11Eh		19Eh
ADCON0	1Fh	ADCON1	9Fh		11Fh		19Fh
General Purpose Register 96 Bytes	20h	General Purpose Register 80 Bytes	A0h	General Purpose Register 80 Bytes	120h	General Purpose Register 80 Bytes	1A0h
			EFh		16Fh		1EFh
		accesses 70h-7Fh	F0h	accesses 70h-7Fh	170h	accesses 70h - 7Fh	1F0h
	7Fh		FFh		17Fh		1FFh

Shaded: Unimplemented data memory locations, read as '0'.
* Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
2: These registers are reserved, maintain these registers clear.

 DS30292C-page 13

2.2.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 2-1.

The Special Function Registers can be classified into two sets: core (CPU) and peripheral. Those registers associated with the core functions are described in detail in this section. Those related to the operation of the peripheral features are described in detail in the peripheral features section.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 0											
00h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
01h	TMR0	Timer0 Module Register								xxxx xxxx	47
02h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
03h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
04h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
05h	PORTA	—	—	PORTA Data Latch when written: PORTA pins when read						--0x 0000	29
06h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
07h	PORTC	PORTC Data Latch when written: PORTC pins when read								xxxx xxxx	33
08h(4)	PORTD	PORTD Data Latch when written: PORTD pins when read								xxxx xxxx	35
09h(4)	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	36
0Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
0Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
0Ch	PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	22
0Dh	PIR2	—	(5)	—	EEIF	BCLIF	—	—	CCP2IF	-r-0 0--0	24
0Eh	TMR1L	Holding register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
0Fh	TMR1H	Holding register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	52
10h	T1CON	—	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	51
11h	TMR2	Timer2 Module Register								0000 0000	55
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	55
13h	SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	70, 73
14h	SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	67
15h	CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	57
16h	CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	57
17h	CCP1CON	—	—	CCP1X	CCP1Y	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	58
18h	RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	96
19h	TXREG	USART Transmit Data Register								0000 0000	99
1Ah	RCREG	USART Receive Data Register								0000 0000	101
1Bh	CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	57
1Ch	CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	57
1Dh	CCP2CON	—	—	CCP2X	CCP2Y	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	58
1Eh	ADRESH	A/D Result Register High Byte								xxxx xxxx	116
1Fh	ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{DONE}$	—	ADON	0000 00-0	111

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.
2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.
3: These registers can be addressed from any bank.
4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.
5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

PIC16F87X

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 1											
80h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
81h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
82h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
83h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
84h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	29
86h	TRISB	PORTB Data Direction Register								1111 1111	31
87h	TRISC	PORTC Data Direction Register								1111 1111	33
88h(4)	TRISD	PORTD Data Direction Register								1111 1111	35
89h(4)	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	37
8Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
8Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
8Ch	PIE1	PSPIE(2)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	21
8Dh	PIE2	—	**(5)**	—	EEIE	BCLIE	—	—	CCP2IE	-r-0 0--0	23
8Eh	PCON	—	—	—	—	—	—	$\overline{POR}$	$\overline{BOR}$	---- --qq	25
8Fh	—	Unimplemented								—	—
90h	—	Unimplemented								—	—
91h	SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	68
92h	PR2	Timer2 Period Register								1111 1111	55
93h	SSPADD	Synchronous Serial Port (I^2C mode) Address Register								0000 0000	73, 74
94h	SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	66
95h	—	Unimplemented								—	—
96h	—	Unimplemented								—	—
97h	—	Unimplemented								—	—
98h	TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	95
99h	SPBRG	Baud Rate Generator Register								0000 0000	97
9Ah	—	Unimplemented								—	—
9Bh	—	Unimplemented								—	—
9Ch	—	Unimplemented								—	—
9Dh	—	Unimplemented								—	—
9Eh	ADRESL	A/D Result Register Low Byte								xxxx xxxx	116
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	0--- 0000	112

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved.
Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.
2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.
3: These registers can be addressed from any bank.
4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.
5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

TABLE 2-1: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Details on page:
Bank 2											
100h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
101h	TMR0	Timer0 Module Register								xxxx xxxx	47
102h(3)	PCL	Program Counter's (PC) Least Significant Byte								0000 0000	26
103h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
104h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
105h	—	Unimplemented								—	—
106h	PORTB	PORTB Data Latch when written: PORTB pins when read								xxxx xxxx	31
107h	—	Unimplemented								—	—
108h	—	Unimplemented								—	—
109h	—	Unimplemented								—	—
10Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
10Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
10Ch	EEDATA	EEPROM Data Register Low Byte								xxxx xxxx	41
10Dh	EEADR	EEPROM Address Register Low Byte								xxxx xxxx	41
10Eh	EEDATH	—	—	EEPROM Data Register High Byte						xxxx xxxx	41
10Fh	EEADRH	—	—	—	EEPROM Address Register High Byte					xxxx xxxx	41
Bank 3											
180h(3)	INDF	Addressing this location uses contents of FSR to address data memory (not a physical register)								0000 0000	27
181h	OPTION_REG	$\overline{RBPU}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	19
182h(3)	PCL	Program Counter (PC) Least Significant Byte								0000 0000	26
183h(3)	STATUS	IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C	0001 1xxx	18
184h(3)	FSR	Indirect Data Memory Address Pointer								xxxx xxxx	27
185h	—	Unimplemented								—	—
186h	TRISB	PORTB Data Direction Register								1111 1111	31
187h	—	Unimplemented								—	—
188h	—	Unimplemented								—	—
189h	—	Unimplemented								—	—
18Ah(1,3)	PCLATH	—	—	—	Write Buffer for the upper 5 bits of the Program Counter					---0 0000	26
18Bh(3)	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	20
18Ch	EECON1	EEPGD	—	—	—	WRERR	WREN	WR	RD	x--- x000	41, 42
18Dh	EECON2	EEPROM Control Register2 (not a physical register)								---- ----	41
18Eh	—	Reserved maintain clear								0000 0000	—
18Fh	—	Reserved maintain clear								0000 0000	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, read as '0', r = reserved. Shaded locations are unimplemented, read as '0'.

Note 1: The upper byte of the program counter is not directly accessible. PCLATH is a holding register for the PC<12:8> whose contents are transferred to the upper byte of the program counter.

2: Bits PSPIE and PSPIF are reserved on PIC16F873/876 devices; always maintain these bits clear.

3: These registers can be addressed from any bank.

4: PORTD, PORTE, TRISD, and TRISE are not physically implemented on PIC16F873/876 devices; read as '0'.

5: PIR2<6> and PIE2<6> are reserved on these devices; always maintain these bits clear.

PIC16F87X

2.2.2.1 STATUS Register

The STATUS register contains the arithmetic status of the ALU, the RESET status and the bank select bits for data memory.

The STATUS register can be the destination for any instruction, as with any other register. If the STATUS register is the destination for an instruction that affects the Z, DC or C bits, then the write to these three bits is disabled. These bits are set or cleared according to the device logic. Furthermore, the $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits are not writable, therefore, the result of an instruction with the STATUS register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the STATUS register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C or DC bits from the STATUS register. For other instructions not affecting any status bits, see the “Instruction Set Summary."

> **Note:** The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ bit, respectively, in subtraction. See the `SUBLW` and `SUBWF` instructions for examples.

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{\text{TO}}$	$\overline{\text{PD}}$	Z	DC	C
bit 7							bit 0

bit 7 **IRP:** Register Bank Select bit (used for indirect addressing)
`1` = Bank 2, 3 (100h - 1FFh)
`0` = Bank 0, 1 (00h - FFh)

bit 6-5 **RP1:RP0**: Register Bank Select bits (used for direct addressing)
`11` = Bank 3 (180h - 1FFh)
`10` = Bank 2 (100h - 17Fh)
`01` = Bank 1 (80h - FFh)
`00` = Bank 0 (00h - 7Fh)
Each bank is 128 bytes

bit 4 **$\overline{\text{TO}}$**: Time-out bit
`1` = After power-up, `CLRWDT` instruction, or `SLEEP` instruction
`0` = A WDT time-out occurred

bit 3 **$\overline{\text{PD}}$**: Power-down bit
`1` = After power-up or by the `CLRWDT` instruction
`0` = By execution of the `SLEEP` instruction

bit 2 **Z**: Zero bit
`1` = The result of an arithmetic or logic operation is zero
`0` = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
(for $\overline{\text{borrow}}$, the polarity is reversed)
`1` = A carry-out from the 4th low order bit of the result occurred
`0` = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{\text{borrow}}$ bit (`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions)
`1` = A carry-out from the Most Significant bit of the result occurred
`0` = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the two’s complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high, or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	’1’ = Bit is set	’0’ = Bit is cleared	x = Bit is unknown

PIC16F87X

2.2.2.3 INTCON Register

The INTCON Register is a readable and writable register, which contains various enable and flag bits for the TMR0 register overflow, RB Port change and External RB0/INT pin interrupts.

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>). User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7 **GIE:** Global Interrupt Enable bit
1 = Enables all unmasked interrupts
0 = Disables all interrupts

bit 6 **PEIE**: Peripheral Interrupt Enable bit
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts

bit 5 **T0IE**: TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 interrupt
0 = Disables the TMR0 interrupt

bit 4 **INTE**: RB0/INT External Interrupt Enable bit
1 = Enables the RB0/INT external interrupt
0 = Disables the RB0/INT external interrupt

bit 3 **RBIE**: RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **T0IF**: TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INTF**: RB0/INT External Interrupt Flag bit
1 = The RB0/INT external interrupt occurred (must be cleared in software)
0 = The RB0/INT external interrupt did not occur

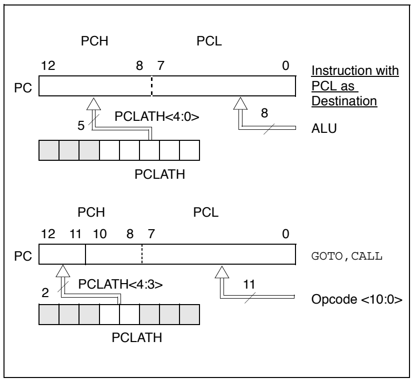
bit 0 **RBIF**: RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state; a mismatch condition will continue to set the bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared (must be cleared in software).
0 = None of the RB7:RB4 pins have changed state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	’1’ = Bit is set	’0’ = Bit is cleared	x = Bit is unknown

2.3 PCL and PCLATH

The program counter (PC) is 13-bits wide. The low byte comes from the PCL register, which is a readable and writable register. The upper bits (PC<12:8>) are not readable, but are indirectly writable through the PCLATH register. On any RESET, the upper bits of the PC will be cleared. Figure 2-5 shows the two situations for the loading of the PC. The upper example in the figure shows how the PC is loaded on a write to PCL (PCLATH<4:0> → PCH). The lower example in the figure shows how the PC is loaded during a `CALL` or `GOTO` instruction (PCLATH<4:3> → PCH).

FIGURE 2-5: LOADING OF PC IN DIFFERENT SITUATIONS



2.3.1 COMPUTED GOTO

A computed `GOTO` is accomplished by adding an offset to the program counter (`ADDWF PCL`). When doing a table read using a computed `GOTO` method, care should be exercised if the table location crosses a PCL memory boundary (each 256 byte block). Refer to the application note, *"Implementing a Table Read"* (AN556).

2.3.2 STACK

The PIC16F87X family has an 8-level deep x 13-bit wide hardware stack. The stack space is not part of either program or data space and the stack pointer is not readable or writable. The PC is PUSHed onto the stack when a `CALL` instruction is executed, or an interrupt causes a branch. The stack is POPed in the event of a `RETURN`,`RETLW` or a `RETFIE` instruction execution. PCLATH is not affected by a PUSH or POP operation.

The stack operates as a circular buffer. This means that after the stack has been PUSHed eight times, the ninth push overwrites the value that was stored from the first push. The tenth push overwrites the second push (and so on).

> **Note 1:** There are no status bits to indicate stack overflow or stack underflow conditions.
>
> **2:** There are no instructions/mnemonics called PUSH or POP. These are actions that occur from the execution of the `CALL`, `RETURN`, `RETLW` and `RETFIE` instructions, or the vectoring to an interrupt address.

2.4 Program Memory Paging

All PIC16F87X devices are capable of addressing a continuous 8K word block of program memory. The `CALL` and `GOTO` instructions provide only 11 bits of address to allow branching within any 2K program memory page. When doing a `CALL` or `GOTO` instruction, the upper 2 bits of the address are provided by PCLATH<4:3>. When doing a `CALL` or `GOTO` instruction, the user must ensure that the page select bits are programmed so that the desired program memory page is addressed. If a return from a `CALL` instruction (or interrupt) is executed, the entire 13-bit PC is popped off the stack. Therefore, manipulation of the PCLATH<4:3> bits is not required for the return instructions (which POPs the address from the stack).

> **Note:** The contents of the PCLATH register are unchanged after a `RETURN` or `RETFIE` instruction is executed. The user must rewrite the contents of the PCLATH register for any subsequent subroutine calls or `GOTO` instructions.

Example 2-1 shows the calling of a subroutine in page 1 of the program memory. This example assumes that PCLATH is saved and restored by the Interrupt Service Routine (if interrupts are used).

EXAMPLE 2-1: CALL OF A SUBROUTINE IN PAGE 1 FROM PAGE 0

```
        ORG 0x500
        BCF PCLATH,4
        BSF PCLATH,3  ;Select page 1
                      ;(800h-FFFh)
        CALL SUB1_P1  ;Call subroutine in
        :             ;page 1 (800h-FFFh)
        :
        ORG 0x900     ;page 1 (800h-FFFh)
SUB1_P1
        :             ;called subroutine
                      ;page 1 (800h-FFFh)
        :
        RETURN        ;return to
                      ;Call subroutine
                      ;in page 0
                      ;(000h-7FFh)
```

2.5 Indirect Addressing, INDF and FSR Registers

The INDF register is not a physical register. Addressing the INDF register will cause indirect addressing.

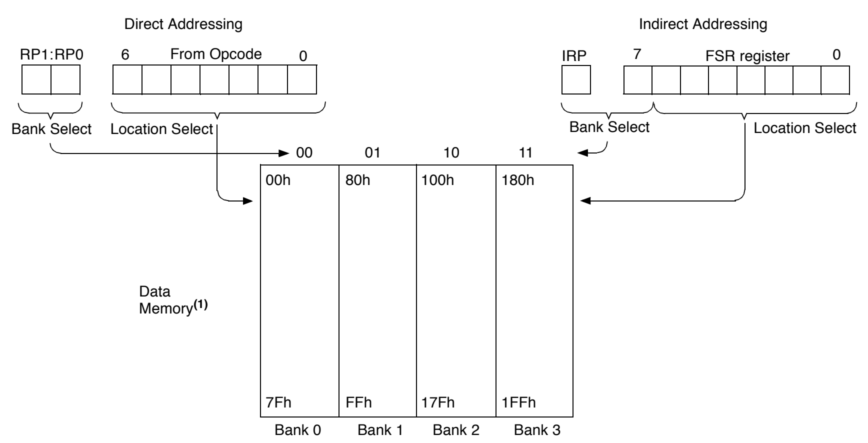
Indirect addressing is possible by using the INDF register. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = '0') will read 00h. Writing to the INDF register indirectly results in a no operation (although status bits may be affected). An effective 9-bit address is obtained by concatenating the 8-bit FSR register and the IRP bit (STATUS<7>), as shown in Figure 2-6.

A simple program to clear RAM locations 20h-2Fh using indirect addressing is shown in Example 2-2.

EXAMPLE 2-2: INDIRECT ADDRESSING

```
         MOVLW  0x20    ;initialize pointer
         MOVWF  FSR     ;to RAM
NEXT     CLRF   INDF    ;clear INDF register
         INCF   FSR,F   ;inc pointer
         BTFSS  FSR,4   ;all done?
         GOTO   NEXT    ;no clear next
CONTINUE
         :              ;yes continue
```

FIGURE 2-6: DIRECT/INDIRECT ADDRESSING



Note 1: For register file map detail, see Figure 2-3.

3.0 I/O PORTS

Some pins for these I/O ports are multiplexed with an alternate function for the peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

Additional information on I/O ports may be found in the PICmicro™ Mid-Range Reference Manual, (DS33023).

3.1 PORTA and the TRISA Register

PORTA is a 6-bit wide, bi-directional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= 1) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISA bit (= 0) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, the value is modified and then written to the port data latch.

Pin RA4 is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open drain output. All other PORTA pins have TTL input levels and full CMOS output drivers.

Other PORTA pins are multiplexed with analog inputs and analog VREF input. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register1).

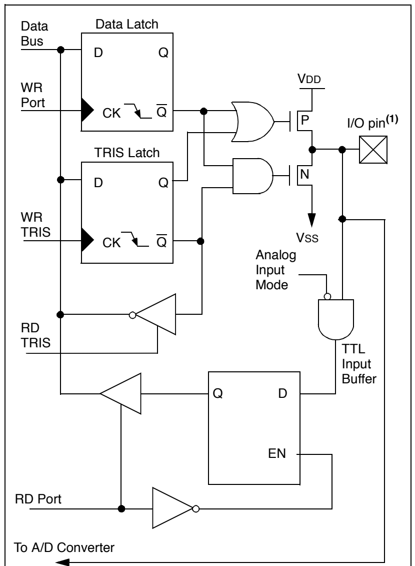
Note:	On a Power-on Reset, these pins are configured as analog inputs and read as '0'.

The TRISA register controls the direction of the RA pins, even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 3-1: INITIALIZING PORTA

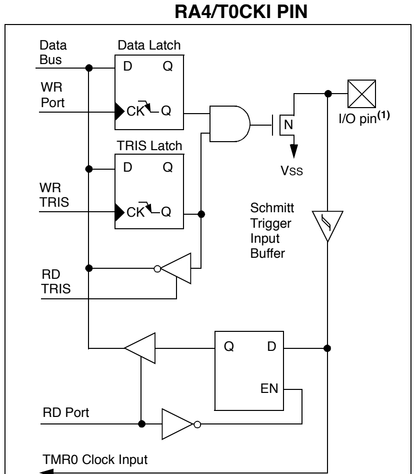
```
BCF     STATUS, RP0  ;
BCF     STATUS, RP1  ; Bank0
CLRF    PORTA        ; Initialize PORTA by
                     ; clearing output
                     ; data latches
BSF     STATUS, RP0  ; Select Bank 1
MOVLW   0x06         ; Configure all pins
MOVWF   ADCON1       ; as digital inputs
MOVLW   0xCF         ; Value used to
                     ; initialize data
                     ; direction
MOVWF   TRISA        ; Set RA<3:0> as inputs
                     ; RA<5:4> as outputs
                     ; TRISA<7:6>are always
                     ; read as '0'.
```

FIGURE 3-1: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS



Note 1: I/O pins have protection diodes to VDD and VSS.

FIGURE 3-2: BLOCK DIAGRAM OF RA4/T0CKI PIN



Note 1: I/O pin has protection diodes to VSS only.

PIC16F87X

TABLE 3-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit0	TTL	Input/output or analog input.
RA1/AN1	bit1	TTL	Input/output or analog input.
RA2/AN2	bit2	TTL	Input/output or analog input.
RA3/AN3/VREF	bit3	TTL	Input/output or analog input or VREF.
RA4/T0CKI	bit4	ST	Input/output or external clock input for Timer0. Output is open drain type.
RA5/$\overline{SS}$/AN4	bit5	TTL	Input/output or slave select input for synchronous serial port or analog input.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 3-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
05h	PORTA	—	—	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
85h	TRISA	—	—	PORTA Data Direction Register						--11 1111	--11 1111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented locations read as '0'.
Shaded cells are not used by PORTA.

Note:	When using the SSP module in SPI Slave mode and $\overline{SS}$ enabled, the A/D converter must be set to one of the following modes, where PCFG3:PCFG0 = 0100,0101, 011x, 1101, 1110, 1111.

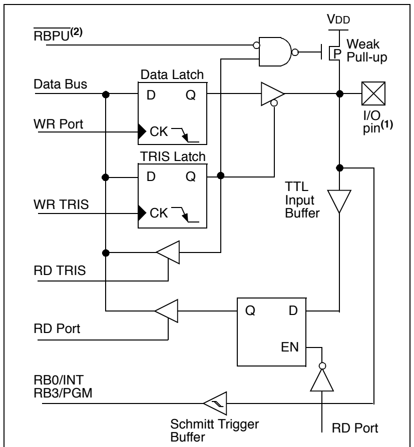
3.2 PORTB and the TRISB Register

PORTB is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= 1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISB bit (= 0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Three pins of PORTB are multiplexed with the Low Voltage Programming function: RB3/PGM, RB6/PGC and RB7/PGD. The alternate functions of these pins are described in the Special Features Section.

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (OPTION_REG<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

FIGURE 3-3: BLOCK DIAGRAM OF RB3:RB0 PINS



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

Four of the PORTB pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The “mismatch” outputs of RB7:RB4 are OR’ed together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON<0>).

This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB. This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

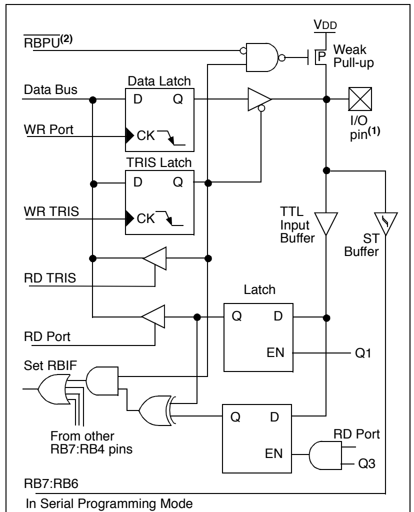
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

This interrupt-on-mismatch feature, together with software configureable pull-ups on these four pins, allow easy interface to a keypad and make it possible for wake-up on key depression. Refer to the Embedded Control Handbook, *“Implementing Wake-up on Key Strokes”* (AN552).

RB0/INT is an external interrupt input pin and is configured using the INTEDG bit (OPTION_REG<6>).

RB0/INT is discussed in detail in Section 12.10.1.

FIGURE 3-4: BLOCK DIAGRAM OF RB7:RB4 PINS



Note 1: I/O pins have diode protection to VDD and VSS.

2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

PIC16F87X

TABLE 3-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT	bit0	TTL/ST(1)	Input/output pin or external interrupt input. Internal software programmable weak pull-up.
RB1	bit1	TTL	Input/output pin. Internal software programmable weak pull-up.
RB2	bit2	TTL	Input/output pin. Internal software programmable weak pull-up.
RB3/PGM(3)	bit3	TTL	Input/output pin or programming pin in LVP mode. Internal software programmable weak pull-up.
RB4	bit4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5	bit5	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB6/PGC	bit6	TTL/ST(2)	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming clock.
RB7/PGD	bit7	TTL/ST(2)	Input/output pin (with interrupt-on-change) or In-Circuit Debugger pin. Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.

2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

3: Low Voltage ICSP Programming (LVP) is enabled by default, which disables the RB3 I/O function. LVP must be disabled to enable RB3 as an I/O pin and allow maximum compatibility to the other 28-pin and 40-pin mid-range devices.

TABLE 3-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
06h, 106h	PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
86h, 186h	TRISB	PORTB Data Direction Register								1111 1111	1111 1111
81h, 181h	OPTION_REG	RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

3.3 PORTC and the TRISC Register

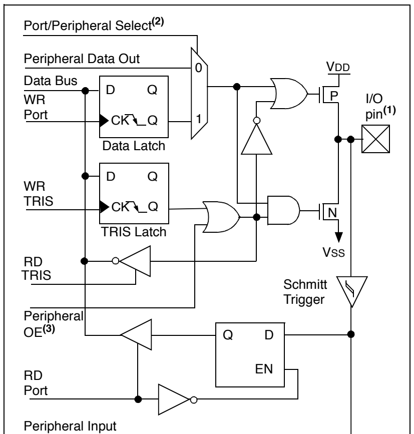
PORTC is an 8-bit wide, bi-directional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= 1) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a Hi-Impedance mode). Clearing a TRISC bit (= 0) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

PORTC is multiplexed with several peripheral functions (Table 3-5). PORTC pins have Schmitt Trigger input buffers.

When the I^2C module is enabled, the PORTC<4:3> pins can be configured with normal I^2C levels, or with SMBus levels by using the CKE bit (SSPSTAT<6>).

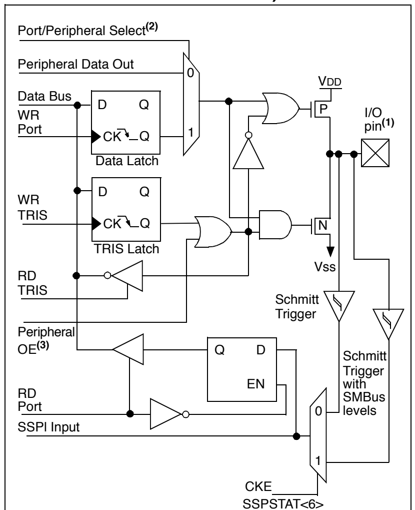
When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. Since the TRIS bit override is in effect while the peripheral is enabled, read-modify-write instructions (`BSF, BCF, XORWF`) with TRISC as destination, should be avoided. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

FIGURE 3-5: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<2:0>, RC<7:5>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral select signal selects between port data and peripheral output.
3: Peripheral OE (output enable) is only activated if peripheral select is active.

FIGURE 3-6: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE) RC<4:3>



Note 1: I/O pins have diode protection to VDD and VSS.
2: Port/Peripheral select signal selects between port data and peripheral output.
3: Peripheral OE (output enable) is only activated if peripheral select is active.

PIC16F87X

TABLE 3-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T1CKI	bit0	ST	Input/output port pin or Timer1 oscillator output/Timer1 clock input.
RC1/T1OSI/CCP2	bit1	ST	Input/output port pin or Timer1 oscillator input or Capture2 input/ Compare2 output/PWM2 output.
RC2/CCP1	bit2	ST	Input/output port pin or Capture1 input/Compare1 output/ PWM1 output.
RC3/SCK/SCL	bit3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit4	ST	RC4 can also be the SPI Data In (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit5	ST	Input/output port pin or Synchronous Serial Port data output.
RC6/TX/CK	bit6	ST	Input/output port pin or USART Asynchronous Transmit or Synchronous Clock.
RC7/RX/DT	bit7	ST	Input/output port pin or USART Asynchronous Receive or Synchronous Data.

Legend: ST = Schmitt Trigger input

TABLE 3-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
07h	PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
87h	TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

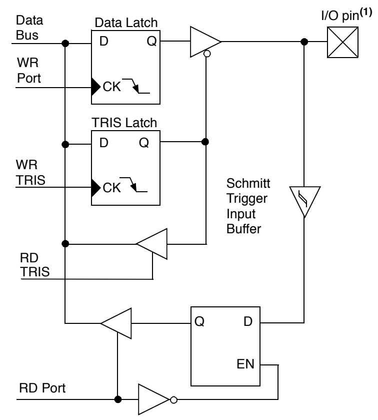
3.4 PORTD and TRISD Registers

PORTD and TRISD are not implemented on the PIC16F873 or PIC16F876.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configureable as an input or output.

PORTD can be configured as an 8-bit wide microprocessor port (parallel slave port) by setting control bit PSPMODE (TRISE<4>). In this mode, the input buffers are TTL.

FIGURE 3-7: PORTD BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 3-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0	bit0	ST/TTL(1)	Input/output port pin or parallel slave port bit0.
RD1/PSP1	bit1	ST/TTL(1)	Input/output port pin or parallel slave port bit1.
RD2/PSP2	bit2	ST/TTL(1)	Input/output port pin or parallel slave port bit2.
RD3/PSP3	bit3	ST/TTL(1)	Input/output port pin or parallel slave port bit3.
RD4/PSP4	bit4	ST/TTL(1)	Input/output port pin or parallel slave port bit4.
RD5/PSP5	bit5	ST/TTL(1)	Input/output port pin or parallel slave port bit5.
RD6/PSP6	bit6	ST/TTL(1)	Input/output port pin or parallel slave port bit6.
RD7/PSP7	bit7	ST/TTL(1)	Input/output port pin or parallel slave port bit7.

Legend: ST = Schmitt Trigger input, TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 3-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
08h	PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
88h	TRISD	PORTD Data Direction Register								1111 1111	1111 1111
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTD.

PIC16F87X

3.5 PORTE and TRISE Register

PORTE and TRISE are not implemented on the PIC16F873 or PIC16F876.

PORTE has three pins (RE0/$\overline{RD}$/AN5, RE1/$\overline{WR}$/AN6, and RE2/$\overline{CS}$/AN7) which are individually configureable as inputs or outputs. These pins have Schmitt Trigger input buffers.

The PORTE pins become the I/O control inputs for the microprocessor port when bit PSPMODE (TRISE<4>) is set. In this mode, the user must make certain that the TRISE<2:0> bits are set, and that the pins are configured as digital inputs. Also ensure that ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

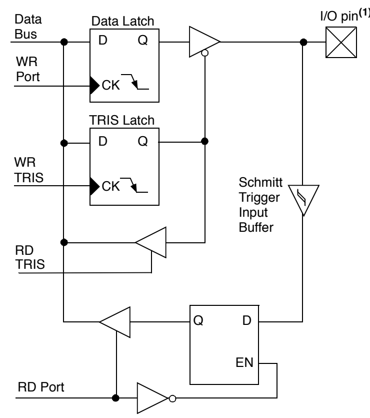
Register 3-1 shows the TRISE register, which also controls the parallel slave port operation.

PORTE pins are multiplexed with analog inputs. When selected for analog input, these pins will read as '0's.

TRISE controls the direction of the RE pins, even when they are being used as analog inputs. The user must make sure to keep the pins configured as inputs when using them as analog inputs.

Note:	On a Power-on Reset, these pins are configured as analog inputs, and read as '0'.

FIGURE 3-8: PORTE BLOCK DIAGRAM (IN I/O PORT MODE)



Note 1: I/O pins have protection diodes to V_{DD} and V_{SS}.

TABLE 3-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AN5	bit0	ST/TTL(1)	I/O port pin or read control input in Parallel Slave Port mode or analog input: $\overline{RD}$ 1 = Idle 0 = Read operation. Contents of PORTD register are output to PORTD I/O pins (if chip selected)
RE1/$\overline{WR}$/AN6	bit1	ST/TTL(1)	I/O port pin or write control input in Parallel Slave Port mode or analog input: $\overline{WR}$ 1 = Idle 0 = Write operation. Value of PORTD I/O pins is latched into PORTD register (if chip selected)
RE2/$\overline{CS}$/AN7	bit2	ST/TTL(1)	I/O port pin or chip select control input in Parallel Slave Port mode or analog input: $\overline{CS}$ 1 = Device is not selected 0 = Device is selected

Legend: ST = Schmitt Trigger input, TTL = TTL input
Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in Parallel Slave Port mode.

TABLE 3-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other RESETS
09h	PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -xxx	---- -uuu
89h	TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE Data Direction Bits			0000 -111	0000 -111
9Fh	ADCON1	ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0	--0- 0000	--0- 0000

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by PORTE.

4.0 DATA EEPROM AND FLASH PROGRAM MEMORY

The Data EEPROM and FLASH Program Memory are readable and writable during normal operation over the entire V_{DD} range. These operations take place on a single byte for Data EEPROM memory and a single word for Program memory. A write operation causes an erase-then-write operation to take place on the specified byte or word. A bulk erase operation may not be issued from user code (which includes removing code protection).

Access to program memory allows for checksum calculation. The values written to program memory do not need to be valid instructions. Therefore, up to 14-bit numbers can be stored in memory for use as calibration parameters, serial numbers, packed 7-bit ASCII, etc. Executing a program memory location containing data that form an invalid instruction, results in the execution of a `NOP` instruction.

The EEPROM Data memory is rated for high erase/write cycles (specification D120). The FLASH program memory is rated much lower (specification D130), because EEPROM data memory can be used to store frequently updated values. An on-chip timer controls the write time and it will vary with voltage and temperature, as well as from chip to chip. Please refer to the specifications for exact limits (specifications D122 and D133).

A byte or word write automatically erases the location and writes the new value (erase before write). Writing to EEPROM data memory does not impact the operation of the device. Writing to program memory will cease the execution of instructions until the write is complete. The program memory cannot be accessed during the write. During the write operation, the oscillator continues to run, the peripherals continue to function and interrupt events will be detected and essentially "queued" until the write is complete. When the write completes, the next instruction in the pipeline is executed and the branch to the interrupt vector will take place, if the interrupt is enabled and occurred during the write.

Read and write access to both memories take place indirectly through a set of Special Function Registers (SFR). The six SFRs used are:

- EEDATA
- EEDATH
- EEADR
- EEADRH
- EECON1
- EECON2

The EEPROM data memory allows byte read and write operations without interfering with the normal operation of the microcontroller. When interfacing to EEPROM data memory, the EEADR register holds the address to be accessed. Depending on the operation, the EEDATA register holds the data to be written, or the data read, at the address in EEADR. The PIC16F873/874 devices have 128 bytes of EEPROM data memory and therefore, require that the MSb of EEADR remain clear. The EEPROM data memory on these devices do not wrap around to 0, i.e., 0x80 in the EEADR does not map to 0x00. The PIC16F876/877 devices have 256 bytes of EEPROM data memory and therefore, uses all 8-bits of the EEADR.

The FLASH program memory allows non-intrusive read access, but write operations cause the device to stop executing instructions, until the write completes. When interfacing to the program memory, the EEADRH:EEADR registers form a two-byte word, which holds the 13-bit address of the memory location being accessed. The register combination of EEDATH:EEDATA holds the 14-bit data for writes, or reflects the value of program memory after a read operation. Just as in EEPROM data memory accesses, the value of the EEADRH:EEADR registers must be within the valid range of program memory, depending on the device: 0000h to 1FFFh for the PIC16F873/874, or 0000h to 3FFFh for the PIC16F876/877. Addresses outside of this range do not wrap around to 0000h (i.e., 4000h does not map to 0000h on the PIC16F877).

4.1 EECON1 and EECON2 Registers

The EECON1 register is the control register for configuring and initiating the access. The EECON2 register is not a physically implemented register, but is used exclusively in the memory write sequence to prevent inadvertent writes.

There are many bits used to control the read and write operations to EEPROM data and FLASH program memory. The EEPGD bit determines if the access will be a program or data memory access. When clear, any subsequent operations will work on the EEPROM data memory. When set, all subsequent operations will operate in the program memory.

Read operations only use one additional bit, RD, which initiates the read operation from the desired memory location. Once this bit is set, the value of the desired memory location will be available in the data registers. This bit cannot be cleared by firmware. It is automatically cleared at the end of the read operation. For EEPROM data memory reads, the data will be available in the EEDATA register in the very next instruction cycle after the RD bit is set. For program memory reads, the data will be loaded into the EEDATH:EEDATA registers, following the second instruction after the RD bit is set.

PIC16F87X

Write operations have two control bits, WR and WREN, and two status bits, WRERR and EEIF. The WREN bit is used to enable or disable the write operation. When WREN is clear, the write operation will be disabled. Therefore, the WREN bit must be set before executing a write operation. The WR bit is used to initiate the write operation. It also is automatically cleared at the end of the write operation. The interrupt flag EEIF is used to determine when the memory write completes. This flag must be cleared in software before setting the WR bit. For EEPROM data memory, once the WREN bit and the WR bit have been set, the desired memory address in EEADR will be erased, followed by a write of the data in EEDATA. This operation takes place in parallel with the microcontroller continuing to execute normally. When the write is complete, the EEIF flag bit will be set. For program memory, once the WREN bit and the WR bit have been set, the microcontroller will cease to execute instructions. The desired memory location pointed to by EEADRH:EEADR will be erased. Then, the data value in EEDATH:EEDATA will be programmed. When complete, the EEIF flag bit will be set and the microcontroller will continue to execute code.

The WRERR bit is used to indicate when the PIC16F87X device has been reset during a write operation. WRERR should be cleared after Power-on Reset. Thereafter, it should be checked on any other RESET. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset, or a WDT Time-out Reset, during normal operation. In these situations, following a RESET, the user should check the WRERR bit and rewrite the memory location, if set. The contents of the data registers, address registers and EEPGD bit are not affected by either $\overline{\text{MCLR}}$ Reset, or WDT Time-out Reset, during normal operation.

REGISTER 4-1: EECON1 REGISTER (ADDRESS 18Ch)

R/W-x	U-0	U-0	U-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	—	—	—	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD**: Program/Data EEPROM Select bit
1 = Accesses program memory
0 = Accesses data memory
(This bit cannot be changed while a read or write operation is in progress)

bit 6-4 **Unimplemented:** Read as '0'

bit 3 **WRERR**: EEPROM Error Flag bit
1 = A write operation is prematurely terminated
(any $\overline{\text{MCLR}}$ Reset or any WDT Reset during normal operation)
0 = The write operation completed

bit 2 **WREN**: EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1 **WR**: Write Control bit
1 = Initiates a write cycle. (The bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD**: Read Control bit
1 = Initiates an EEPROM read. (RD is cleared in hardware. The RD bit can only be set (not cleared) in software.)
0 = Does not initiate an EEPROM read

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	’1’ = Bit is set	’0’ = Bit is cleared	x = Bit is unknown

4.2 Reading the EEPROM Data Memory

Reading EEPROM data memory only requires that the desired address to access be written to the EEADR register and clear the EEPGD bit. After the RD bit is set, data will be available in the EEDATA register on the very next instruction cycle. EEDATA will hold this value until another read operation is initiated or until it is written by firmware.

The steps to reading the EEPROM data memory are:

1. Write the address to EEDATA. Make sure that the address is not larger than the memory size of the PIC16F87X device.
2. Clear the EEPGD bit to point to EEPROM data memory.
3. Set the RD bit to start the read operation.
4. Read the data from the EEDATA register.

EXAMPLE 4-1: EEPROM DATA READ

```
BSF    STATUS, RP1    ;
BCF    STATUS, RP0    ;Bank 2
MOVF   ADDR, W        ;Write address
MOVWF  EEADR          ;to read from
BSF    STATUS, RP0    ;Bank 3
BCF    EECON1, EEPGD  ;Point to Data memory
BSF    EECON1, RD     ;Start read operation
BCF    STATUS, RP0    ;Bank 2
MOVF   EEDATA, W      ;W = EEDATA
```

4.3 Writing to the EEPROM Data Memory

There are many steps in writing to the EEPROM data memory. Both address and data values must be written to the SFRs. The EEPGD bit must be cleared, and the WREN bit must be set, to enable writes. The WREN bit should be kept clear at all times, except when writing to the EEPROM data. The WR bit can only be set if the WREN bit was set in a previous operation, i.e., they both cannot be set in the same operation. The WREN bit should then be cleared by firmware after the write. Clearing the WREN bit before the write actually completes will not terminate the write in progress.

Writes to EEPROM data memory must also be prefaced with a special sequence of instructions, that prevent inadvertent write operations. This is a sequence of five instructions that must be executed without interruptions. The firmware should verify that a write is not in progress, before starting another cycle.

The steps to write to EEPROM data memory are:

1. If step 10 is not implemented, check the WR bit to see if a write is in progress.
2. Write the address to EEADR. Make sure that the address is not larger than the memory size of the PIC16F87X device.
3. Write the 8-bit data value to be programmed in the EEDATA register.
4. Clear the EEPGD bit to point to EEPROM data memory.
5. Set the WREN bit to enable program operations.
6. Disable interrupts (if enabled).
7. Execute the special five instruction sequence:
   - Write 55h to EECON2 in two steps (first to W, then to EECON2)
   - Write AAh to EECON2 in two steps (first to W, then to EECON2)
   - Set the WR bit
8. Enable interrupts (if using interrupts).
9. Clear the WREN bit to disable program operations.
10. At the completion of the write cycle, the WR bit is cleared and the EEIF interrupt flag bit is set. (EEIF must be cleared by firmware.) If step 1 is not implemented, then firmware should check for EEIF to be set, or WR to clear, to indicate the end of the program cycle.

EXAMPLE 4-2: EEPROM DATA WRITE

```
BSF    STATUS, RP1    ;
BSF    STATUS, RP0    ;Bank 3
BTFSC  EECON1, WR     ;Wait for
GOTO   $-1            ;write to finish
BCF    STATUS, RP0    ;Bank 2
MOVF   ADDR, W        ;Address to
MOVWF  EEADR          ;write to
MOVF   VALUE, W       ;Data to
MOVWF  EEDATA         ;write
BSF    STATUS, RP0    ;Bank 3
BCF    EECON1, EEPGD  ;Point to Data memory
BSF    EECON1, WREN   ;Enable writes
                      ;Only disable interrupts
BCF    INTCON, GIE    ;if already enabled,
                      ;otherwise discard
MOVLW  0x55           ;Write 55h to
MOVWF  EECON2         ;EECON2
MOVLW  0xAA           ;Write AAh to
MOVWF  EECON2         ;EECON2
BSF    EECON1, WR     ;Start write operation
                      ;Only enable interrupts
BSF    INTCON, GIE    ;if using interrupts,
                      ;otherwise discard
BCF    EECON1, WREN   ;Disable writes
```

11.0 ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) Converter module has five inputs for the 28-pin devices and eight for the other devices.

The analog input charges a sample and hold capacitor. The output of the sample and hold capacitor is the input into the converter. The converter then generates a digital result of this analog level via successive approximation. The A/D conversion of the analog input signal results in a corresponding 10-bit digital number. The A/D module has high and low voltage reference input that is software selectable to some combination of VDD, VSS, RA2, or RA3.

The A/D converter has a unique feature of being able to operate while the device is in SLEEP mode. To operate in SLEEP, the A/D clock must be derived from the A/D's internal RC oscillator.

The A/D module has four registers. These registers are:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register0 (ADCON0)
- A/D Control Register1 (ADCON1)

The ADCON0 register, shown in Register 11-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 11-2, configures the functions of the port pins. The port pins can be configured as analog inputs (RA3 can also be the voltage reference), or as digital I/O.

Additional information on using the A/D module can be found in the PICmicro™ Mid-Range MCU Family Reference Manual (DS33023).

REGISTER 11-1: ADCON0 REGISTER (ADDRESS: 1Fh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0:** A/D Conversion Clock Select bits
00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (clock derived from the internal A/D module RC oscillator)

bit 5-3 **CHS2:CHS0**: Analog Channel Select bits
000 = channel 0, (RA0/AN0)
001 = channel 1, (RA1/AN1)
010 = channel 2, (RA2/AN2)
011 = channel 3, (RA3/AN3)
100 = channel 4, (RA5/AN4)
101 = channel 5, (RE0/AN5)(1)
110 = channel 6, (RE1/AN6)(1)
111 = channel 7, (RE2/AN7)(1)

bit 2 **GO/$\overline{\text{DONE}}$:** A/D Conversion Status bit
If ADON = 1:
1 = A/D conversion in progress (setting this bit starts the A/D conversion)
0 = A/D conversion not in progress (this bit is automatically cleared by hardware when the A/D conversion is complete)

bit 1 **Unimplemented**: Read as '0'

bit 0 **ADON**: A/D On bit
1 = A/D converter module is operating
0 = A/D converter module is shut-off and consumes no operating current

Note 1: These channels are not available on PIC16F873/876 devices.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

PIC16F87X

REGISTER 11-2: ADCON1 REGISTER (ADDRESS 9Fh)

U-0	U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. 6 Most Significant bits of ADRESH are read as '0'.
0 = Left justified. 6 Least Significant bits of ADRESL are read as '0'.

bit 6-4 **Unimplemented:** Read as '0'

bit 3-0 **PCFG3:PCFG0**: A/D Port Configuration Control bits:

PCFG3: PCFG0	AN7(1) RE2	AN6(1) RE1	AN5(1) RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0	VREF+	VREF-	CHAN/ Refs(2)
0000	A	A	A	A	A	A	A	A	VDD	VSS	8/0
0001	A	A	A	A	VREF+	A	A	A	RA3	VSS	7/1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5/0
0011	D	D	D	A	VREF+	A	A	A	RA3	VSS	4/1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3/0
0101	D	D	D	D	VREF+	D	A	A	RA3	VSS	2/1
011x	D	D	D	D	D	D	D	D	VDD	VSS	0/0
1000	A	A	A	A	VREF+	VREF-	A	A	RA3	RA2	6/2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6/0
1010	D	D	A	A	VREF+	A	A	A	RA3	VSS	5/1
1011	D	D	A	A	VREF+	VREF-	A	A	RA3	RA2	4/2
1100	D	D	D	A	VREF+	VREF-	A	A	RA3	RA2	3/2
1101	D	D	D	D	VREF+	VREF-	A	A	RA3	RA2	2/2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1/0
1111	D	D	D	D	VREF+	VREF-	D	A	RA3	RA2	1/2

A = Analog input D = Digital I/O

Note 1: These channels are not available on PIC16F873/876 devices.
2: This column indicates the number of analog channels available as A/D inputs and the number of analog channels used as voltage reference inputs.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

The ADRESH:ADRESL registers contain the 10-bit result of the A/D conversion. When the A/D conversion is complete, the result is loaded into this A/D result register pair, the GO/$\overline{\text{DONE}}$ bit (ADCON0<2>) is cleared and the A/D interrupt flag bit ADIF is set. The block diagram of the A/D module is shown in Figure 11-1.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as inputs.

To determine sample time, see Section 11.1. After this acquisition time has elapsed, the A/D conversion can be started.

10.0 ADDRESSABLE UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full duplex asynchronous system that can communicate with peripheral devices such as CRT terminals and personal computers, or it can be configured as a half duplex synchronous system that can communicate with peripheral devices such as A/D or D/A integrated circuits, serial EEPROMs etc.

The USART can be configured in the following modes:

- Asynchronous (full duplex)
- Synchronous - Master (half duplex)
- Synchronous - Slave (half duplex)

Bit SPEN (RCSTA<7>) and bits TRISC<7:6> have to be set in order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter.

The USART module also has a multi-processor communication capability using 9-bit address detection.

REGISTER 10-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER (ADDRESS 98h)

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9**: 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN**: Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in SYNC mode.

bit 4 **SYNC**: USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **BRGH**: High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode

bit 1 **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data, can be parity bit

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 10-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER (ADDRESS 18h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RC7/RX/DT and RC6/TX/CK pins as serial port pins)
0 = Serial port disabled

bit 6 **RX9**: 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN**: Single Receive Enable bit
Asynchronous mode:
Don't care
Synchronous mode - master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode - slave:
Don't care

bit 4 **CREN**: Continuous Receive Enable bit
Asynchronous mode:
1 = Enables continuous receive
0 = Disables continuous receive
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and load of the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received, and ninth bit can be used as parity bit

bit 2 **FERR**: Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receive next valid byte)
0 = No framing error

bit 1 **OERR**: Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data (can be parity bit, but must be calculated by user firmware)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown