Coin Sorting and Counting Machine

By

Ray Anthony C. Reyes
Mark Phillip P. Tan
Timothy Jude O. Torralba

A Design Report Submitted to the School of Electrical Engineering, Electronics and Communications Engineering, and Computer Engineering in Partial Fulfillment of the Requirements for the Degree

Bachelor of Science in Computer Engineering

Mapua Institute of Technology
November 2008

Mapua Institute of Technology
School of EE – ECE – CoE

This is to certify that we have supervised the preparation of and read the design report prepared by **Ray Anthony C. Reyes, Mark Phillip P. Tan, Timothy Jude O. Torralba** entitled **Coin Sorting and Counting Machine** and that the said report has been submitted for final examination by the Oral Examination Committee.

Ayumang
Engr. Analyn Yumang
Reader

Engr. Jocelyn F. Villaverde
Design Adviser

As members of the Oral Examination Committee, we certify that we have examined this design report, presented before the committee on **Nov. 25, 2008**, and hereby recommend that it be accepted as fulfillment of the design requirement for the degree in **Bachelor of Science in Computer Engineering**.

Engr. Isagani V. Villamor
Panel Member

Engr. Ernesto M. Vergara
Panel Member

msantos
Engr. Joyce M. Santos
Chairman

This design report is hereby approved and accepted by the School of Electrical Engineering, Electronics and Communications Engineering, and Computer Engineering as fulfillment of the design requirement for the degree in **Bachelor of Science in Computer Engineering**.

Felicito S Caluyo
Dr. Felicito S. Caluyo
Dean, School of EE – ECE - CoE

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The design was entitled Coin Sorting and Counting Machine for it is to sort and count the Philippine coin denomination except for the 5 and 10 centavo coins. This was done to be able to help different instituitions that deal with large number of coins in their daily operation. Some of these are banks, churches, charitable instituitions and the transport sector. The sorting part of the design is basically mechanical as it sorts the coins through their physical size then counts them as they pass through the phototransistor. After performing some tests with the Coin Sorting and Counting Machine, the group could say that an almost accurate and reliable machine was created as manifested by the results it made.

Keywords: denomination, coin slot, rotating plate, microcontroller

Chapter 1

DESIGN BACKGROUND AND INTRODUCTION

Design Setting

In the Philippines, the first coins were used in 1861. Until now, people still use coins in their everyday lives. Ther are those that deal with coins everyday like banks, tranport groups, casinos, charitable instituitions and the like. Daily, banks produce and receive coins which they deliver to other establishments to supply their need for coins. Charitbale instituitions like churches gather coins from donations during mass. And in public transport sector, coins are widely used whether as fares or change. These establishments count and sort coins everyday. Without the use of machines that can handle large operations regarding coins, an establishment needs to hire an extra employee just to deal with coins or just count them manually where it could take too much of his time which could be used for other work.

The design was developed to help the trade and industry in counting coins faster and easier. Using phototransistors, the design counts faster and more accurate because it is not sensitive to visible light. A PIC Microcontroller was used to determine the total number of coins, total amount per denomination and the total amount of all coins. A three-way switch is also used to stop the count every 50 or 100 coins or to continuously count.

It is surprising how people manage to sort and count coins especially in large quantity. The group thought of a design that would try to help and solve the problem of counting coins of large quantity. With the design, the group can help the trade and industry by providing a simple yet effective way of dealing with large quantity of coins and saving precious time which can be put to other work.

Statement of the Problem

Persons who manually sort and count coins of large quantity usually take too much of their time and may refrain themselves from doing other tasks. There are available coin sorters and counters in the market but are very expensive. The main problem of the study is to find a way to count and sort Philippine coins of different denominations in less time and with accuracy.

Objective of the Design

The general objective of the design was to develop a coin sorting machine that sorts and counts Philippine coins. The other specific objectives were as follows:

1. Use the PIC Microcontroller to program the machine to stop every 50 or 100 coins counted.
2. Provide more accuracy and reliability in counting coins.
3. Create a design with less cost and more simple.

4. Sort a variety of coins in less time.
5. To contribute something beneficial to the trade and industry.

Significance of the Study

This study is an improvement of the existing coin sorter being used in the country.

The group wanted to share the design project to owners of establishments and small businesses by letting them try to use this kind of technology in their everyday dealings with sorting and counting coins. The proposed design will be affordable to ordinary people and can be applicable to various industries like banking, charitable institutions and etc. The design is also implemental because the equipment and materials to be used are locally available. The users also have an option to limit the coins to be sorted.

To students, the design will be beneficial since it develops their creativity by improvising on their own way on how to sort and count different denominations of Philippine coin. The design would be helpful to students especially those who keep spare coins and store them in a piggy bank. Students will have an easier task of counting coins.

The school especially the treasury department can directly benefit from the design since it can use the machine for its daily operation. A good example will be during the enrollment period when treasury personnel are handling large amount of coins. They can use the design to sort or count coins according to denomination.

Conceptual Framework

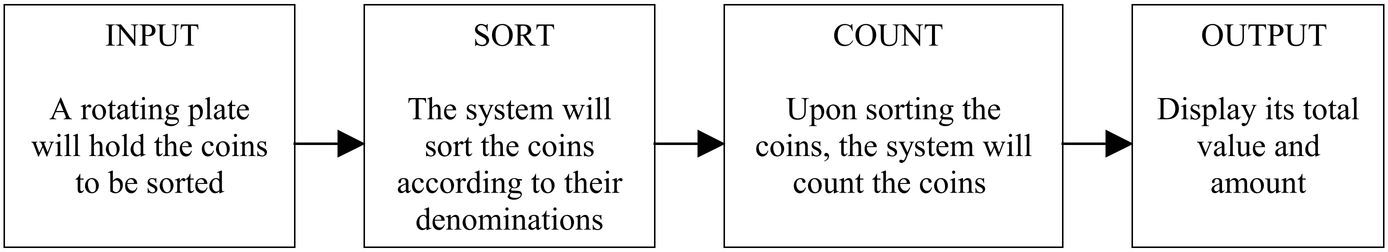

INPUT
A rotating plate will hold the coins to be sorted
SORT
The system will sort the coins according to their denominations
COUNT
Upon sorting the coins, the system will count the coins
OUTPUT
Display its total value and amount

Figure 1.1 Conceptual Framework

The design will sort the coins according to its denomination and size, starting from the smallest to the largest coin. The designers thought that this is the best way to sort coins. A coin slot will be used to distinguish the difference between a 5 peso coin and a 10 peso coin since they are similar in size. A sensor will be used to count the sum total of the coins and display the value to a Liquid Crystal Display (LCD) panel.

Scope and Delimitations

The features and capabilities of the design are the scope of the study:

1. The design can sort and count Philippine coin denominations, namely, 25 centavo coin, 1 peso coin, 5 peso coin and 10 peso coin only.

2. The rotating plate can only handle a certain weight for it to function properly, that is, approximately 30-40 coins at a time.
3. The number of pieces and corresponding value will be displayed using a Liquified Crystal Display (LCD).
4. The grand total also updates whenever any of the reset buttons is pressed. The sensor will count anything that passes through and count it.
5. The design can also be fitted with a battery to save the data and retain its value, but cannot run the motor with just a battery.

There are the delimitations which the design cannot perform.

1. The machine cannot accept deformed coins and cannot recognize whether the coin is fake or genuine. For example, if a token was placed instead of a coin in the machine, the system would still sort it as long as it is the same size as the coins.
2. The display can only handle up to 4 digit numbers.
3. 5 centavo and 10 centavo coins can be sorted but not with perfect accuracy.
4. The coins once sorted and counted will be placed in a holding container randomly and not stacked.

Definition of Terms

AC – (Alternating Current) is a current that varies sinusoidally with time. (Alexander and Sadiku, 2004)

AC Motor – a type of motor that uses alternating current (AC). (Young and Freedman, 2000)

Accuracy – a measured value – that is, how close it is likely to be to the true value. (Young and Freedman, 2000)

Capacitance – is the ratio of the charge on one plate of a capacitor to the voltage difference between the two plates. (Alexander and Sadiku, 2004)

Capacitor – is a passive element designed to store energy in its electric field. (Alexander and Sadiku, 2004)

Ceramic Capacitor – a capacitor that is light in weight, stable and its change in temperature is predictable. (Alexander and Sadiku, 2004)

Coin – is usually a piece of hard material, usually metal or a metallic material, usually in the shape of a disc, and most often issued by a government. (Merriam Webster Dictionary)

Counter - a program that counts and typically displays how many. (Grout, 2007)

Crystal Oscillator – an oscillator that uses a quartz crystal to generate a frequency. Such devices generally output a fixed frequency, but some can be controlled by a tuning voltage over a small range. (Purdie, 1998-2001)

DC – (Direct Current) is a current that remains constant with time. (Alexander and Sadiku, 2004)

DC Motor – a type of motor that is driven by direct current (DC). (Young and Freedman, 2000)

Denomination – a proper description of a currency amount, usually for coins or banknotes. (Merriam Webster Dictionary)

Diode - comprises a section of N-type material bonded to a section of P-type material, with electrodes on each end. (Harris, 1998-2008)

Electrolytic Capacitor – a type of capacitor that produces very high capacitance. (Alexander and Sadiku, 2004)

Flowchart – is a graphical representation of a process such as a manufacturing operation or computer operation, indicating the various steps that are taken as the product moves along the production line or the problem moves through the computer. (Boilot, Gleason and Horn, 1979)

Infrared - refers to energy in the region of the electromagnetic radiation spectrum at wavelengths longer than those of visible light, but shorter than those of radio waves. (Tech Target: The IT Media ROI Experts, 2008)

LCD – (Liquid Crystal Display) a low-power flat-panel display used in many laptop computers, calculators and digital watches, made up of a liquid crystal that is sandwiched between layers of glass or plastic and becomes opaque when electric current passes through it. The contrast between the opaque and transparent areas forms visible characters. (Castellano, 2005)

LED – (Light Emitting Diode) is a semiconductor device that emits visible light when an electric current passes through it. (Harris, 1998-2008)

Microcontroller - are "**embedded**" inside some other device (often a consumer product) so that they can control the features or actions of the product. (Grout, 2007)

Oscillator – a mechanical or electronic device that works on the principles of oscillation: a periodic fluctuation between two things based on changes in energy. (Alexander and Sadiku, 2004)

PCB – (Printed Circuit Board) used to mechanically support and electrically connect electronic components using conductive pathways, or traces, etched from copper sheets laminated onto a non-conductive substrate. (Grout, 2007)

Photosensor - is an electronic component that detects the presence of visible light, infrared transmission (IR), and/or ultraviolet (UV) energy. (Tech Target: The IT Media ROI Experts, 2008)

PIC Microcontroller – a microcontroller popular with developers and hobbyists alike due to its low cost, wide availability, large user base, extensive collection of application notes, availability of low cost or free development tools, and serial programming capability. (Grout, 2007)

Power Supply – a device or system that supplies electrical or other types of energy to an output load or group of loads is called a power supply unit or PSU. (Floyd, 2003)

Rectifier – an electrical device that converts alternating current (AC) to direct current (DC). (Floyd, 2003)

Resistance – the ratio of voltage to current for a particular conductor. (Young and Freedman, 2000)

Resistor – a circuit device made to have a specific value of resistance between its ends. (Young and Freedman, 2000)

Transformer – a device that transfers electrical energy from one circuit to another through inductively coupled electrical conductors. (Floyd, 2003)

Transistor - regulates current or voltage flow and acts as a switch or gate for electronic signals. (Tech Target: The IT Media ROI Experts, 2008)

Voltage Regulator – is an installable module that senses a computer's microprocessor voltage requirements and ensures that the correct voltage is maintained. (Tech Target: The IT Media ROI Experts, 2008)

Chapter 2

REVIEW OF RELATED LITERATURE AND STUDIES

How does the Philippine coin sorter and counter works

The Philippine coin sorter works by sorting the coins by their diameter and counts the coins using a photo-sensor. The device uses a geared motor that drives the propeller which is connected to a rotating plate which feeds the coins to the sorting arm. The sorting arm then sorts the coins by their diameter and drops them to the corresponding conveyor. The count is then stored on a computer system for it to be totaled. (Lactaeaon et al., 2006)

The Philippine coin sorter is a simple but a not so reliable type of coin sorting machine. Firstly, it uses a geared motor that turns only in one direction. This can be a cause of problem in case the coins get jammed in the rotating plate. The device also uses a sorting arm which is quite long and is not always efficient in sorting because many factors interfere. Some factors are the angle of inclination of the sorting arm, the friction in the sorting arm and the length of the sorting arm. This is why it takes longer for the Philippine coin sorter to sort coins. The display is bulky since it is stored in a computer rather than in an LCD.

Coin sorting apparatus with rotating disc stationary guide plate for sorting coins by their different diameter

An article by Schwartz (October 4, 1988) about a coin counter talks about sorting coins by their diameter. It uses a rotating plate which are filled with holes that catches coin upon insertion. Behind the plate are reference holes designed according to the different diameters of the coin to be sorted. The coin rotates in the rotating plate until it finds a hole that matches its size.

Sorting coins according to their diameter is a simple and yet effective way of sorting. It only uses a rotating plate and a reference plate which can be easily made out of cheap materials. The only downside of this method is when coins get stacked above one another. A smaller coin stacked above a bigger coin can fall in the bigger coin's hole and end up in the wrong place. This can cause some errors in sorting.

Coin sorter apparatus and method utilizing coin thickness as a discriminating parameter

Another article by Rasmussen (October 1, 1985) talks about sorting coins by their thickness. According to the article, the device uses sets of rotating disks which carry the coin into a region wherein it could fit. The first set of disks is positioned so that whenever the coins touch the disks, it would be moved to the second set of disks. The second set of disks are properly placed to match a certain thickness of a certain coin. If a coin is of the right thickness, the coin will pass through and get sorted.

This method of sorting is somewhat not that reliable. It can only be applicable to foreign coins where the thickness vary greatly. Philippine coins are almost of the same thickness. Sorting them accordingly using this method can impose a lot of problems. Secondly, this method requires precision in placing the disks. One miscalculation and the disks can be either too thin or too thick.

Coin Counter

Boland in his article (July 8, 1986) explains a method in coin counting. Coins are counted by using denomination codes that are stored in a code disc. Coins fall upon a rotating table that is designed to catch and hold coins of different denominations at different positions.

The method of counting uses a sensor which stores code in a disc to determine the denomination of a coin. This is an efficient way of counting coins since it is also very unlikely to fail because it uses codes for each kind of coin. The only downside of this method is the cost of materials that are going to be used. Code sensors are very expensive and are available in specialty shops only. The method is somewhat similar to bar coding which also uses a sensor in detecting the bars in every code.

The Philippine Peso Coins (www.bsp.gov.ph)

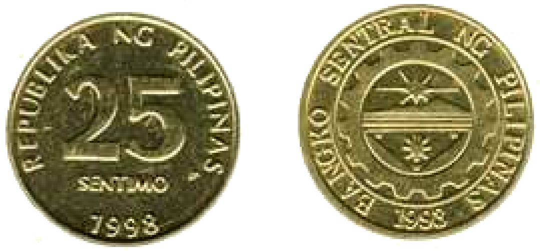


Figure 2.1 A sample of 25 centavo coin

25 Centavos

Composition: 65% Copper, 35% Zinc

Diameter: 20.0mm

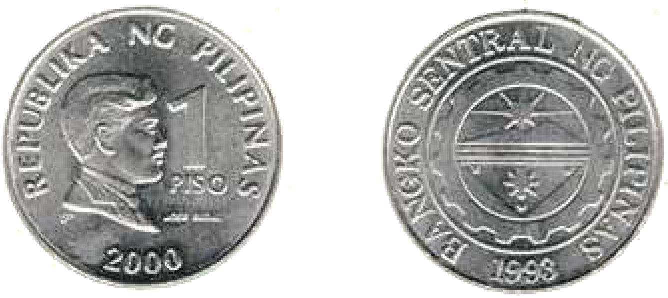


Figure 2.2 A sample of 1 Peso coin

1 Peso

Composition: 75% Copper, 25% Nickel

Diameter: 24.0mm

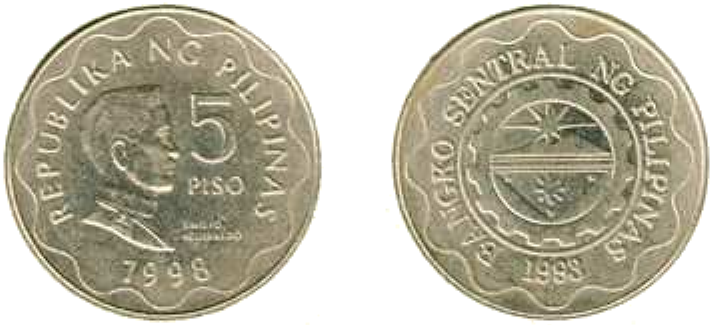


Figure 2.3 A sample of 5 Peso coin

5 Peso

Composition: 70% Copper; 24.5% Zinc; 5.5% Nickel

Diameter: 27.0mm

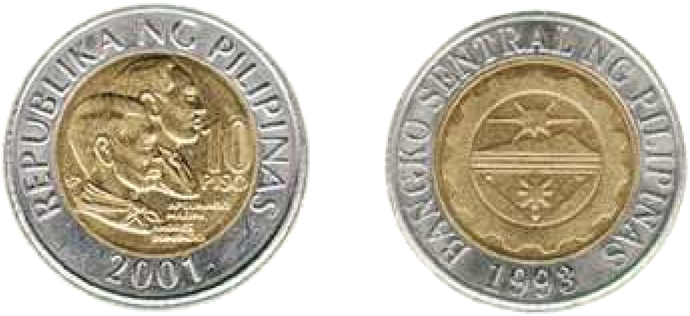


Figure 2.4 A sample of 10 Peso coin

10 Peso

Composition: Ring: 75% Copper; 25% Nickel

Core: 92% Copper; 6% Aluminum; 2% Nickel

Diameter: 26.5mm

Mag-Nif Roll Master CLXX Coin Sorter/Counter **(Retrieved November 11, 2008 from www.factory-express.com)**

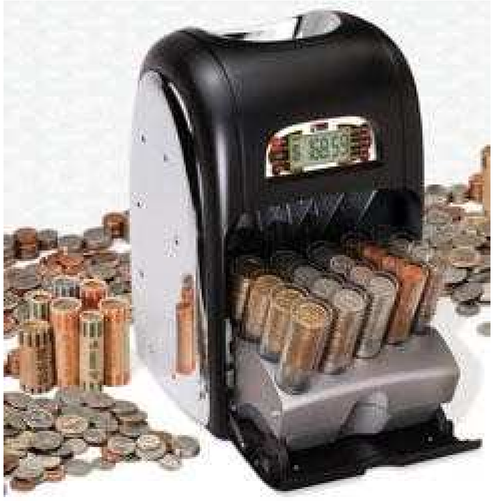


Figure 2.5 Mag-Nif Roll Master CLXX Coin Sorter/Counter full stack

Figure 2.5 shows the Mag-Nif Roll Master CLXX Coin Sorter/Counter. It sorts coins directly into preformed paper wrappers that can be placed into twenty coin tubes. With its durable design and chrome plated stainless steel parts, the Roll Master CLXX can sort thousands of coins in just minutes. The LCD digital display offers many functions. You can keep a total and a running total or grand total from the start. The bank's capacity is $170.00 but the grand total feature continues to count up to $999.99, so you can keep track of a small or large dollar value. You may also subtract dollar values in the total by single coins, by a roll of coins or by an entire denomination of coins. The Roll Master CLXX is equipped with a convenient overflow trap so that you do not have to stop when one coin tube is full or all four coin tubes of one denomination are full. It will continue to sort coins until you are finished. Any excess coins will simply spill into the overflow tray. The Roll Master CLXX is built for speed and accuracy; it includes patented sorting technology and patent pending anti jamming features. A unique feature of the Roll Master CLXX allows full access to the entire mechanism for easy maintenance without the use of any tools. There is an automatic shut off when you access the coins. The Roll Master CLXX is powered by an AC adaptor and two "AA" batteries. (Source, www.factory-express.com)

The Roll Master CLXX is indeed a very good coin sorting and counting machine. Unfortunately, it is only available in other countries and the cost is way higher and it only sorts US coins.

Chapter 3

DESIGN METHODOLOGY AND PROCEDURES

Design Methodology

The design is a constructive research since it is to develop a system to be used as a solution to a problem that needs to be solved. The problem is to create a way to sort and count coins faster than a man can do. The counter and sorter should also be more accurate and easier than manual sorting and counting.

The group researched on existing coin sorter machines as reference on how to develop a prototype that would be accurate and fast and that would observe the accuracy of the techniques the existing designs used. Upon observation, the group tested the different techniques used by existing or recent counters. They tried to develop also new ways and improvements of existing techniques. They had performed several tests on how to efficiently and accurately sort and count coins of different denominations. The group first tested a similar technique done by previous studies which use a sorting arm with the exact holes of every coin that are arranged from the smallest to the largest coins. Two to three holes of each coin are placed simultaneously to increase its reliability in sorting coins. The group also found some problems on how to sort the 5 and 10 peso coins since they are the same in size with a very minimal difference in their thickness. The other test was to use a revolving plate which assures that only a single coin will be sorted in a time, but to be fast, the group decided to create holes in succession. Another way was to sort the coins by their weight but the technology was quite expensive for the group. By doing some research, the group came up with the best way to sort the 5 and 10 peso coins and that was to use a coin slot which uses a sample to compare its magnetic property to the one to be tested.

Design Procedure

The design was composed of the input, sorting, counting and output sections. Included in the input section were the input switches and the funnel that would be used for input of coins. The sorting part was consisted of the revolving plates with holes fitted for philippine denomination coins. These plates were operated by an AC motor for it to revolve. Included in this part was the passage way of each coin to the containers. The material sensor which was used to sort five peso coins from ten peso coins was also included in this part. The infrared sensors located at each passage way of coins is the counting part of the machine. The LCD display served as the output part of the system. A microcontroller and power supply circuit controlled the whole operation of all sections of the prototype.

a. Hardware Design

The following are the procedures in creating the prototype:

1. Prepare the needed materials such as the different kinds of input

switches, LCD display, power supply, acrylic plastic for the body of the prototype, etc.

2. Create all the needed circuits which include the microcontroller circuit and the power supply circuit.
3. Create the body of the prototype and insert the input switches needed in the input section. Connect the input switches to the microcontroller circuit. Use the funnel as input section where the coins will be placed. This is the top part of the prototype.
4. Create the sorting part of the design. This includes the 2 plates with holes fitted for Philippine coin denominations, which serve as the sorter, and the bridges or passage way of the coins to their containers upon sorting, and the material sensor for five peso and ten peso coins.
5. Prepare the AC motor circuit connection and the power supply; then connect it to the sorter. Check if the motor moves the sorter well.
6. For the counting section, place the sensors along the passage way of each coins. Place them opposite each other, in a way, the LED is aligned to the infrared. Connect this sensors to the microcontroller circuit.
7. Connect the LED display to the microcontroller circuit and place in front of the prototype. This is the output section of the prototype.

8. Prepare the program and burn it in the microcontroller. This step is done simultaneously with the hardware part.
9. Test all the sections of the design. Do troubleshooting as needed upon seeing the results of the test.

1. Block Diagram

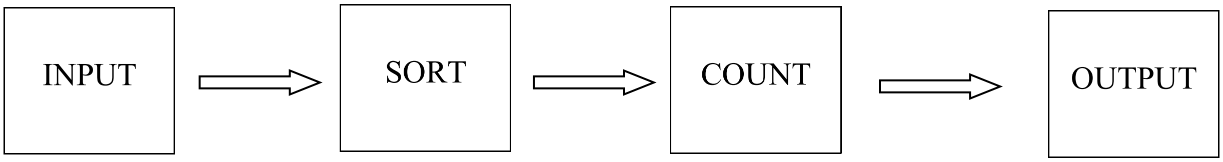


Figure 3.1 Block Diagram of the Design

The design can be divided into input section, sorting, counting and the output part. The input section is the part where the coins will be stored for sorting. Included in this section are the input switches which are the start/resume button, stop button, page button and the reset buttons for each coin denomination. After the input, the stored coins will now be sorted according to denominations in the sorting part of the design. From the sorting section, the coins will now be counted and compute for their equivalent value, in the counting section of the design. Then the results will now be displayed in the LCD, which is the output section of the design.

2. **Schematic Diagram**

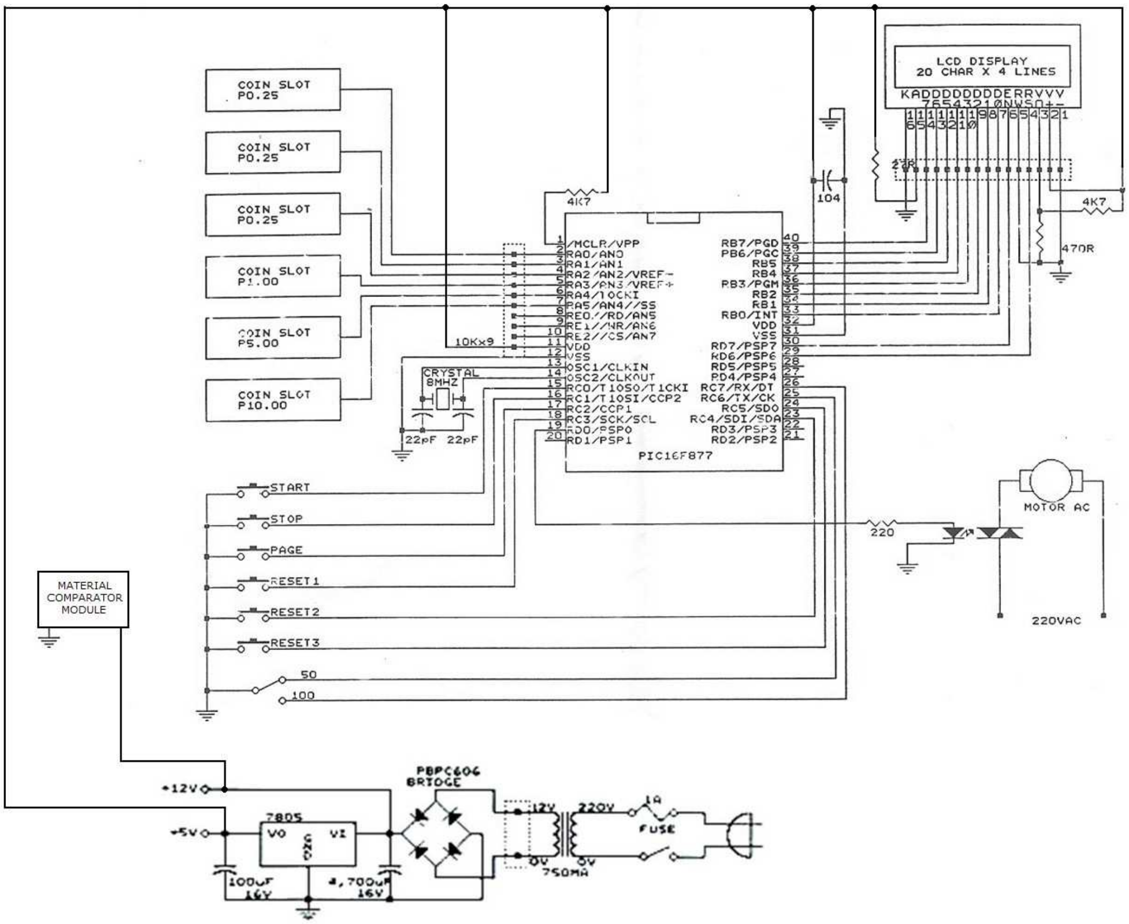


Figure 3.2 Circuit Diagram of the Design

3. List of Materials

Materials	Unit Price (in Php)
4 x 20 LCD	1,900.00
PICI6F877	530.00
40 PINS IC Socket	8.00
7805 Voltage Regulator	15.00
Heat Sink	20.00
10 K Array Resistor	12.00
2 PINS Terminal Block	12.00
W106 Rectifier Diode	10.00
1000µF /16v Elect. Capacitor	6.00
100 µF/25V Elect. Capacitor	3.00
8Mhz Crystal	50.00
22PF Ceramic Capacitor	2.00
Assorted ¼ w resistor	18.00
8 PINS connector	37.00
105 multi layer cap	1.00
Power Supply Module	150.00
1Amp – Transformer	170.00
4700 µF/ 25 V Electrical Capacitor	38.00
PBPC Rectifier Diode	25.00
3 PINS Terminal Block	15.00
IR Sensor	39.00
Rocker Switch	25.00
Fuse Holder w/ fuse	10.00
Camlock	38.00
Hinges	18.00
Mini Push Button	10.00
Acrylic Plastic	1,000.00
AC Cord	48.00
Coin Shoot	1,200.00
AC Motor	600.00

Figure 3.3 List of Materials Used

b. Software Design

1. System Flowchart or Algorithm

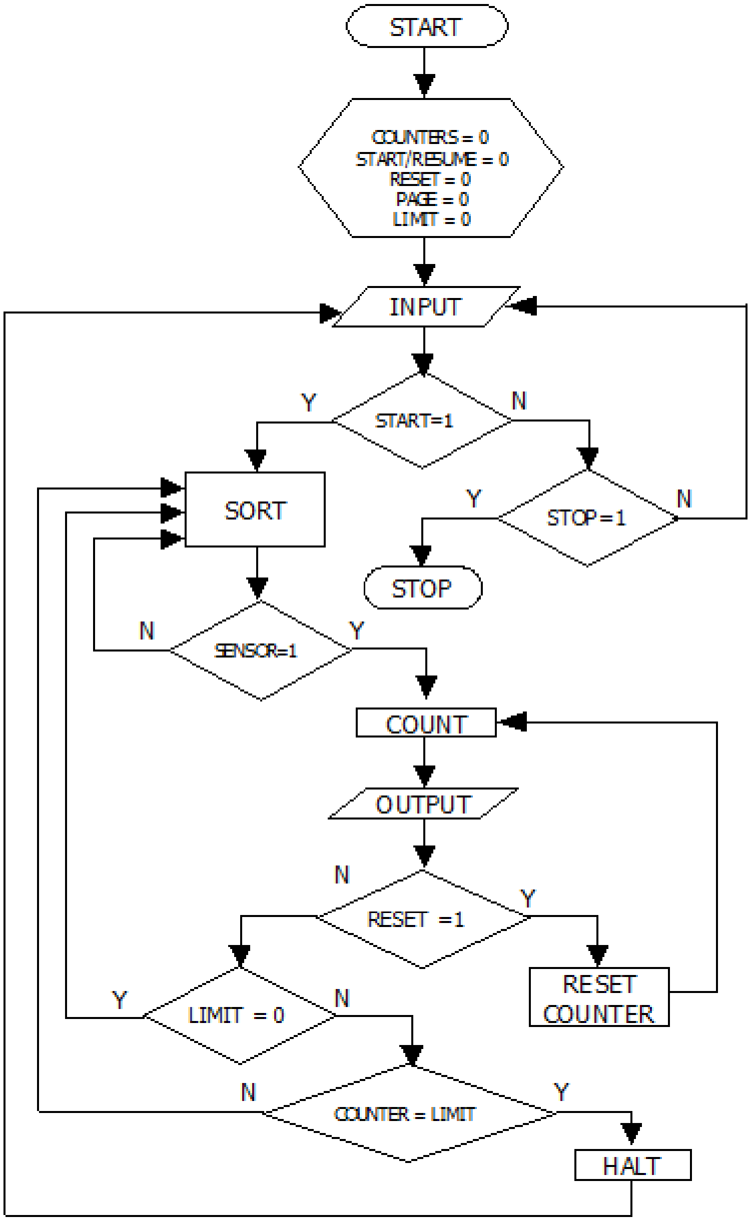


Figure 3.4 System Flowchart

The system starts by initializing all the variables needed such as the input buttons and the counters and the variables in the program. Then it can now accept input from the input buttons and coins from the funnel. If the start button is pressed, the machine will start to sort the coins inside it. Then it will count the sorted coins with the use of sensor. If a coin passes through the sensor, it will be counted by incrementing its specific counter in the program. Then the display output will be updated every time there will be a count process. It will then check if a reset button is already pressed. If any reset button is pressed, the counter which the reset button is pertaining to, will be reset to 0 and update the output in the LCD display. Then the program will check if the limit switch is set to zero or not. If it is set to 0, the machine will continue to sort until the stop button is pressed. If there is a limit, the program will stop the system only if any counter reaches the specified limit. It will continue to sort unless any counter will reach the limit. When the system is stopped or in halt, it will wait for input of start button to resume.

c. Prototype Development

In creating the design, the mechanical part and the program is done at the same time. The sorting portion is the mechanical part of the design. The technique used in sorting process is by the size of the coins and the material of the coins for five and ten peso coin. For the counting portion of the design, infrared phototransistor sensors are used to count the coins.

For the input section, a funnel made of plastic is used to drop coins to the sorter. This is to limit the number of coins and the weight that will be put in the design. In the sorting portion, an AC motor is used to run a revolving plate with holes with the size enough to hold any coin, which will rotate on another plate that contains holes exactly fitted to Philippine coins. These two plates do the coin sorting process. For the five and ten peso coins, since they have the same size, a material sensor is used to separate them. This sensor checks if the input coin has the same material with the sample. In the design, the material sensor uses the five peso coin as the sample. This will accept five peso coin and will reject any other coins that will be inserted to it, which is the ten peso coin in the design.

For the counting portion of the design, infrared phototransistor sensors are put to every end of bridge or tunnel of every coin coming from the sorter. This sensor is used because it is not sensitive to room lighting. This sensor gives a value of logic 1 every time something passes through it. The signal will be processed by the microcontroller to increment the count and compute the equivalent value of the coin. Then these values will be output to a LCD Display of the design. The LCD contains the number of coins counted by the sensor and processor, the equivalent values, and the grand total of all the coins sorted by the machine.

Chapter 4

TESTING, PRESENTATION AND INTERPRETATION OF DATA

The testing of the design made was very critical since it involved money and required much accuracy and reliability. The results would show if the machine had been successful in its objectives and could help the trade and industry. The Coin Sorting and Counting Machine was tested using 100 pieces of each 25 centavo, 1 peso, 5 peso, and 10 peso coins. The 25 centavo coins and 1 peso coins were tested through their physical size while the 5 peso coins and 10 peso coins were compared using a coin slot due to their similarity in size. The coin slot used a 5 peso coin as its sample.

Test conducted for 25 centavo coins

The first test conducted was meant to determine if 25 centavo coins would be properly sorted and counted by the machine.

The first step conducted was to set the limit to 50 pieces. In this case, the motor will automatically stop once it reaches the limit. 100 pieces of 25 centavo coins were placed in the rotating plate. The waiting time was until the coins were counted and reached the limit. After completing three trials, the limit was then set to 100 pieces then the same steps were performed.

The test was timed and monitored in three trials enough to simulate the behavior of the machine.

25 c	**Limit 50**			**Limit 100**		
Trials	Machine Count	Actual Count	Time	Machine Count	Actual Count	Time
trial 1	50	50	58 sec	100	100	81sec
trial 2	50	50	51 sec	99	100	90 sec
trial 3	50	50	53 sec	100	100	79 sec

Table 4.1. Testing for 25 centavo coins

Table 4.1 shows the machine count and the actual count for the testing of 25 centavo coins. Initially, 100 pieces of coins were sorted and counted. The limit was first set to 50 and showed no error in three trials. The time to sort 50 coins of 25 centavos averaged 54 seconds. Then the limit was set to 100, the first and last trial showed no error while the second trial showed that the machine count was only 99 which means that two coins passed the sensor at the same time. The average time to sort 100 pieces of 25 centavo coins is 83.33 seconds.

Test conducted for 1 Peso coins

The purpose of this test was to know if the holes in the rotating plate would exactly fit 1 peso coin and could actually sort it through its physical size.

To sort 100 pieces of 1 peso coin, the limit was set to 50 for the first three trials then to 100 for last three trials as well. After, the sample coins were placed, the machine was turned on. The rotating plate started when the start button was pressed. The motor stopped when it reached the set limit.

P1	**Limit 50**			**Limit 100**		
Trials	Machine Count	Actual Count	Time	Machine Count	Actual Count	Time
trial 1	51	51	56 sec	100	100	87 sec
trial 2	50	50	48 sec	100	100	84 sec
trial 3	51	51	53 sec	100	100	77 sec

Table 4.2. Testing for 1 Peso coins

Table 4.2 presents the results of test conducted for 1 peso coins. The limit was first set to 50; then 100 coins were placed for sorting. The results showed that both trial 1 and trial 3 exceeded the limit of 50 but also showed that the machine and actual count tallied the same number. This means that the machine could not control any coin that passed through the sensor after the motor stopped, but would still count any coin that followed. The average time to sort 1 Peso coins is 52.3. Then the limit was set to 100. It was noticed that there were no errors in the three trials. It would take 83 seconds to sort 100 pieces of 1 Peso coin.

Test conducted for 5 Peso coin

The purpose of the test was to know if the coin slot would work properly in sorting the same sizes of coins. The coin slot was used to distinguish the 5 peso from the 10 peso coin.

Before testing the actual design, the coin slot was tested first. The slot was placed with a 5 peso coin as a sample. Upon placing the sample coin, the machine was then tested by placing a handful of 5 peso coins on the rotating plate at a time since placing the entire 100 pieces of coins would be too heavy for the plates and the motor.

P5	**Limit 50**			**Limit 100**		
Trials	Machine Count	Actual Count	Time	Machine Count	Actual Count	Time
trial 1	51	51	58 sec	100	100	87 sec
trial 2	51	51	63 sec	99	100	124 sec
trial 3	51	51	60 sec	100	100	86 sec

Table 4.3. Testing for 5 Peso coins

As can be gleaned in Table 4.3, all trials have the same number of machine count. Though it exceeded the limit that was set, the machine still provided accuracy satisfactory enough because there was no much discrepancy between the actual count and the machine count. This happened when large quantity of coins was placed in the machine at the same time that it sorted continuously. However when exact quantity was placed, the machine sorted it accurately. The machine could sort 50 pieces of coins in just 1 minute. Upon setting the limit to 100, an error happened on the second trial when the machine counted only 99 pieces when it should have been 100 pieces. The error had occurred probably because 2 coins overlapped when they passed through the sensor so they were counted as a single coin. The time needed to sort 100 pieces of coins is 99 seconds.

Test conducted for 10 Peso coins

The purpose of this test was to sort 5 and 10 peso coins using a coin slot and 5 peso as a sample. Once the magnetic property of the sample is not met, the slot will reject the coin and sort it as 10 peso coin.

In testing the 10 peso coin, the limit switch was first set to 50 then the coins were placed on the revolving plate. After testing 50 pieces, the limit switch was then set to 100 pieces then the coins were again placed. The waiting time was until the limit was reached.

P10	**Limit 50**			**Limit 100**		
Trials	Machine Count	Actual Count	Time	Machine Count	Actual Count	Time
trial 1	50	50	56 sec	99	100	86 sec
trial 2	51	51	58 sec	100	100	90 sec
trial 3	51	51	62 sec	100	100	88 sec

Table 4.4 Testing for 10 Peso coins

Table 4.4 shows the results of the test for the 10 peso coins. Trial 1 had exact number of coins when the limit was set to 50 and both trial 2 and trial 3 exceeded the limit by 1 but could still be acceptable because the machine has no control over stopping the coins that will follow after the limit was reached and the motor stopped. What it can do is to still count the coin the will pass through the sensor. Sorting 50 pieces of 10 peso coins requires almost a minute (58.7 seconds). Upon setting the limit to 100 coins, an error occurred only in the first trial when the machine had counted only 99 coins when it should have been a hundred. 88 seconds was needed to sort 100 pieces of 10 peso coins.

Test conducted for mixed coins

The purpose of this test was to determine the behavior of the machine once different coins were placed at the same time and how fast it could sort and count different denomination of coins.

The first step was to place all coins to be used in a plastic or any container then mix the coins to ensure random sampling. The limit was set to 50 and the waiting time was until all the denominations reached the limit. The same step was done upon switching the limit to 100.

It was assumed that a handful of coins should be placed on the plate at a time so that it could still perform with accuracy and within the capability of the plates and motor.

	25 c		P1		P5		P10		
Trials	Machine Count	Actual Count	Machine Count	Actual Count	Machine Count	Actual Count	Machine Count	Actual Count	Time
trial 1	101	100	99	100	99	100	100	100	10 min 36 sec
trial 2	100	99	101	100	100	100	100	100	11 min 04 sec
trial 3	100	99	99	101	100	100	99	100	10 min 57 sec

For the testing of mixed coins, the limit was set to infinite to determine if all 400 coins would be sorted accordingly to their denominations. The results relied greatly on the quantity of coins placed in the rotating plates that sort the coins. For this test, a large quantity of coins was placed to know the maximum number of coins that the machine can still count with accuracy. As the results showed, the machine could not perform accurately once large quantity was placed at the same time. In trial 1, only the 10 peso coins were sorted and counted properly. Other coin counters showed that they were less than or more than those of the required value. The error occurred probably when the coins were simultaneously sorted and they have passed through sensor at the same time. In trial 2, both the 5 peso and 10 peso coins were counted properly. And in the last trial, the 1 peso coin exceeded because one 25 centavo coin was sorted as 1 peso coin. In all the trials, the time required was 10 to 11 minutes to sort 4 denominations of coins which were 100 pieces each.

Test for Manual Testing

This test was done to compare whether the machine could be more efficient in sorting and counting coins than a person who would do the sorting and counting manually.

The test was done by making 5 different persons sort and count a combination of all coins 100 pieces per denomination. They were given the samples then let them do the test. Their performance time was recorded.

They were asked to manually sort and count the coins without any knowledge of the quantity of the coins.

Trials	Time
trial 1	12 min 05 sec
trial 2	15 min 13 sec
trial 3	13 min 32 sec
trial 4	11 min 41 sec
trial 5	11 min 58 sec

Table 4.6. Manual Testing

Table 4.6 shows the results of the five-trial testing to determine which between machine counting and manual counting was more efficient. The data obtained proved that machine counting was more efficient from manual counting.

Chapter 5

CONCLUSION AND RECOMMENDATION

Conclusion

After performing a number of tests and trials on how the Coin Sorting and Counting Machine performed, it can be concluded that a nearly accurate and reliable machine was created as proven by the results taken. Though a few errors had occurred in certain trials, the percentage of error was minimal and negligible. The machine can perform its best if the quantity of coins placed is not too massive or heavy because the behavior of coins depends greatly on the amount of quantity placed simultaneously in the machine.

The design was also implemented in the best ways possible by using a phototransistor as its sensors and a PIC microcontroller. The phototransistor was used because it is not affected by visible light and it triggers when its beam is cut. The microcontroller was also able to stop the motor once the limit is set.

The sorting mechanism is simple in a way that it sorts the coins by using a plate fitted with holes as exact as the size of each coin and another plate that revolves that gets exactly one coin at a time. Upon revolving, the coins will be sorted if it fits the holes intended for each denomination.

Likewise, it was observed that in less than a minute, the machine could sort up to 50 coins of each single denomination. With this ratio, the machine performs in an acceptable rate knowing that it is only a prototype which sorts and counts Philippine coin denominations. With the machine performing satisfactorily, it can be concluded that the group succeeded in its attempt to design a sorting and counting machine that can be used by sectors such as banks, charities, church, among others. The machine designed by the research team could be a significant contribution to the trade and industry.

Recommendation

The research team recommends the following improvements:

1) Replace the revolving plate with a stainless steel so that it can handle large quantity of coins and that smaller coins which jam in between the plates can be avoided;
2) A stronger motor should be used so that it can handle the weight of the plates and the coins;
3) Each sorting arm can also be fitted with an individual coin slot so that it will compare every coin that passes through it. In this way, the design can be more accurate;
4) The sorting arm can also be provided with something that can lessen the speed of the coins and ensure that they will pass through the sensors flat on the surface;
5) A trapping mechanism can also be installed to determine fake coins and segregate deformed coins.

BIBLIOGRAPHY

Alexander C.K., & Sadiku M.N.O. (2004). *Fundamentals of Electric Circuits* (2nd ed.). New York, NY: McGraw-Hill.

Bangko Sentral ng Pilipinas (2005). *BSP Notes and Coins.* Retrieved November 9, 2008 from http://www.bsp.gov.ph.

Boilot M.H., Gleason G.M., & Horn W.L. (1979). *Essentials of Flowcharting* (5th ed.). University of Michigan: W.C. Brown Co.

Castellano J.A. (2005) *LIQUID GOLD, The Story of Liquid Crystal Displays and the Creation of an Industry.* Singapore: World Scientific Publishing Co. Pte. Ltd.

Factory Express (2008) *Business Machine & Supply Direct.* Retrieved November 8, 2008 from http://www.factory-express.com.

Floyd T.L. (2003). *Electronics Fundamentals: Circuits, Devices and Applications* (6th ed.). Upper Saddle River, NJ: Prentice Hall.

Grout I. (2007). *Digital Systems Design with FPGAS.* Ireland: University of Limerick.

Harris T. (1998-2008). How Stuff Works. *How Light Emitting Diodes Work*. Retrieved November 8, 2008, from http://www.howstuffworks.com.

Lactaeaon MP.A., Manansala DJ.D., Racelis MR.S., & Victorio S.R. (2006). *Enhancement of the Philippine Coin Sorter and Counter System with the Use of PIC Microcontroller.*

Merriam Webster Dictionary (2004).

Purdie I.C. (1998-2001). Electronic Tutorials. *Amateur Radio Tutorial* . Retrieved November 8, 2008, from http://www.eletronic-tutorials.com.

Tech Target (2008). *The IT Media ROI Experts.* Retrieved November 8, 2008, from http://www.techtarget.com.

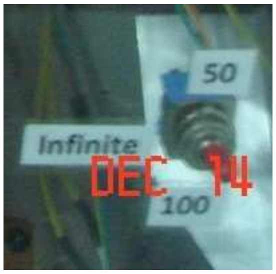
Young H.D., & Freedman R.A. (2000). *University Physics* (10th ed.). Singapore: Pearson Education Asia Pte Ltd.

Appendix A

User's Manual

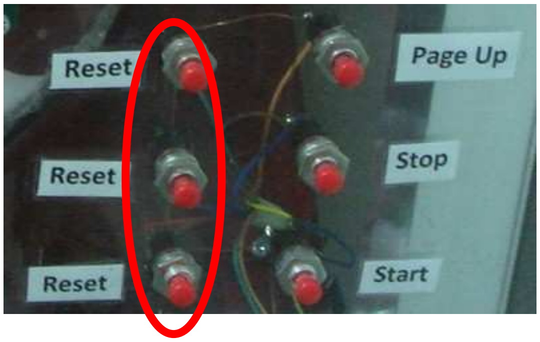
1. Plug the power cord into a 220v source then turn on the main power switch located at the lower left side of the machine.

2. Set the limit switch to the desired number of coins.



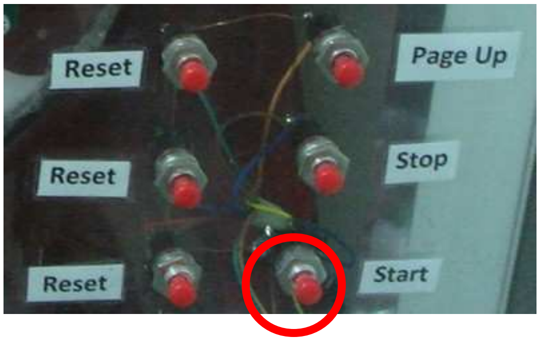
- ❖ Pressing the switch up sets the limit to 50 coins.
- ❖ Pressing the switch down sets the limit to 100 coins.
- ❖ Placing the switch in a neutral position sets the limit to infinite.

3. Press the Reset button for each line to reset the values for each denomination. Press the Page Up to switch between lines.



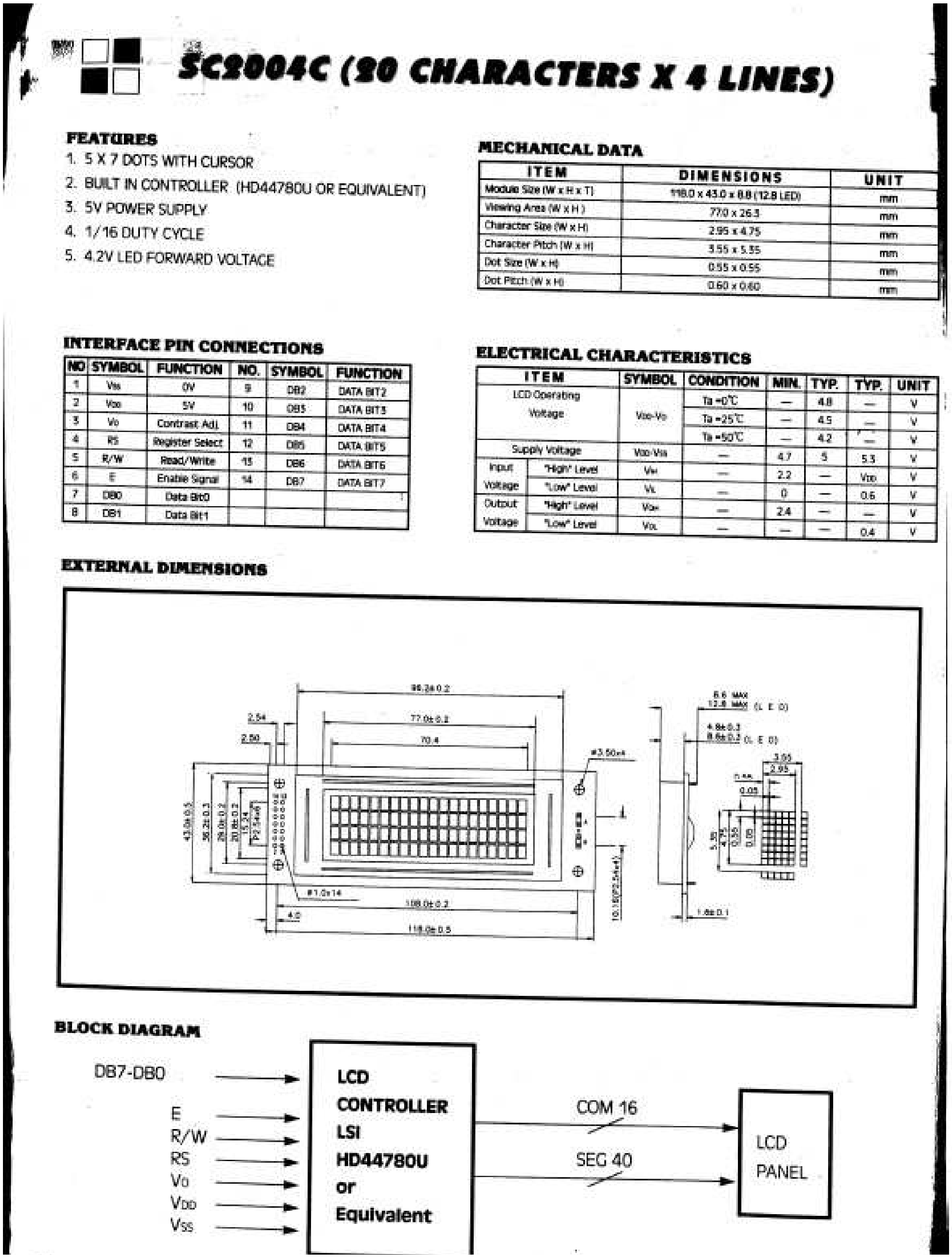
4. Put a handful of coins in the coin tube.

5. Press the Start button to begin sorting and counting.



APPENDIX B

LCD Module Data Sheet



SC2004C (20 CHARACTERS X 4 LINES)

FEATURES

1. 5 X 7 DOTS WITH CURSOR
2. BUILT IN CONTROLLER (HD44780U OR EQUIVALENT)
3. 5V POWER SUPPLY
4. 1/16 DUTY CYCLE
5. 4.2V LED FORWARD VOLTAGE

MECHANICAL DATA

ITEM	DIMENSIONS	UNIT
Module Size (W x H x T)	118.0 x 43.0 x 8.8 (12.8 LED)	mm
Viewing Area (W x H)	77.0 x 26.3	mm
Character Size (W x H)	2.95 x 4.75	mm
Character Pitch (W x H)	3.55 x 5.35	mm
Dot Size (W x H)	0.55 x 0.55	mm
Dot Pitch (W x H)	0.60 x 0.60	mm

INTERFACE PIN CONNECTIONS

NO	SYMBOL	FUNCTION	NO.	SYMBOL	FUNCTION
1	V_{SS}	0V	9	DB2	DATA BIT2
2	V_{DD}	5V	10	DB3	DATA BIT3
3	V_0	Contrast Adj.	11	DB4	DATA BIT4
4	RS	Register Select	12	DB5	DATA BIT5
5	R/W	Read/Write	13	DB6	DATA BIT6
6	E	Enable Signal	14	DB7	DATA BIT7
7	DB0	Data Bit0			
8	DB1	Data Bit1			

ELECTRICAL CHARACTERISTICS

ITEM		SYMBOL	CONDITION	MIN.	TYP.	TYP.	UNIT
LCD Operating Voltage		V_{DD}-V_0	Ta =0℃	—	4.8	—	V
			Ta =25℃	—	4.5	—	V
			Ta =50℃	—	4.2	—	V
Supply Voltage		V_{DD}-V_{SS}	—	4.7	5	5.3	V
Input Voltage	"High" Level	V_{IH}	—	2.2	—	V_{DD}	V
	"Low" Level	V_{IL}	—	0	—	0.6	V
Output Voltage	"High" Level	V_{OH}	—	2.4	—	—	V
	"Low" Level	V_{OL}	—	—	—	0.4	V

EXTERNAL DIMENSIONS

BLOCK DIAGRAM

■ Environmental Conditions

Item	Operating		Non-operating		Remarks
	Min	Max	Min	Max	
Ambient temperature (Ta)	0℃	50℃	-20℃	70℃	Normal temperature type
	-20℃	70℃	-30℃	80℃	Extended temperature type
Vibration	—	0.5G	—	2G	G=9.8 m/s² XYZ directions
Shock	—	3G	—	50G	
Corrosion gas	No corrosion gas				

*SOME GRAPHIC MODULES ARE ONLY AVAILABLE WITH -10℃~60℃

■ Optical Characteristics

The following data are only for reference, please check the details in our product specification.

● TN Type

Item	Symbol	Condition	Min	Typ	Max	Unit	Note
Viewing angle	φ2-φ1	K= 1.4	20	—	—	Deg.	3&4
Contrast ratio	K	φ=25° θ=0°	—	3	—	—	2
Response time (Rise)	tr	φ=25° θ=0°	—	200	250	ms	1
Response time (Fall)	tf	φ=25° θ=0°	—	200	300	ms	1

● STN Type

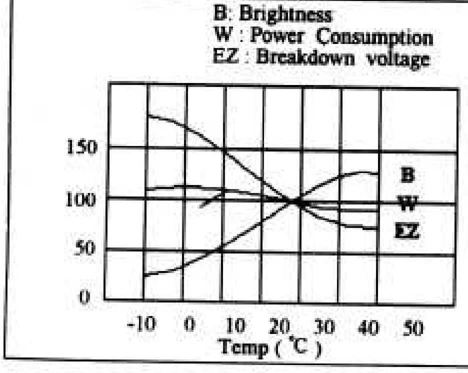
Item	Symbol	Condition	Min	Typ	Max	Unit	Note
Viewing angle	φ2-φ1	K= 1.4	40	—	—	Deg.	3&4
Contrast ratio	K	φ=10° θ=0°	3	—	—	—	2
Response time (Rise)	tr	φ=10° θ=0°	—	200	300	ms	1
Response time (Fall)	tf	φ=10° θ=0°	—	250	350	ms	1

● FSTN Type

Item	Symbol	Condition	Min	Typ	Max	Unit	Note
Viewing angle	φ2-φ1	K= 1.4	50	—	—	Deg.	3&4
Contrast ratio	K	φ=10° θ=0°	5	—	—	—	2
Response time (Rise)	tr	φ=10° θ=0°	—	200	300	ms	1
Response time (Fall)	tf	φ=10° θ=0°	—	250	350	ms	1

■ Backlight Characteristics

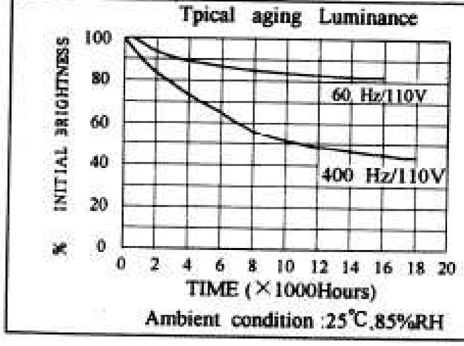
● CCFL backlight

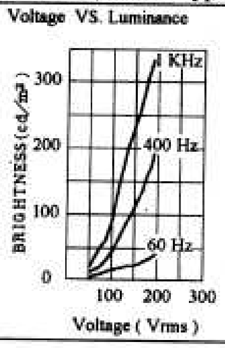


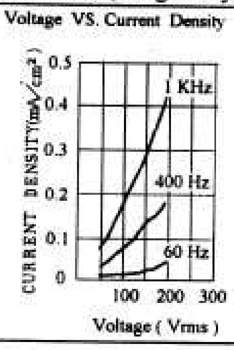
- ●Luminous intensity: 400 cd/㎡
- ●Operating voltage: 1500 Vrms AC Max.
- ●Power consumption: ~1.7W (CCFL only)
- ●Life : 15k~20k hours

● EL Backlight

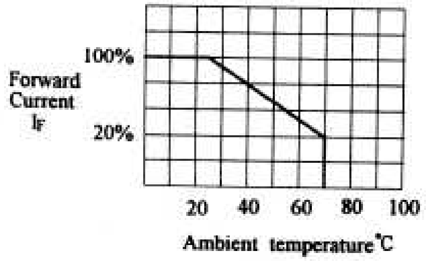
- ●Luminous intensity: 60 cd/㎡
- ●Driving condition: 150 Vrms AC Max. 50~1000 Hz
- ●Power consumption: ~0.17 mA/c㎡
- ●Life : 2k hours (normal type), 8k hours (long life type)







● LED Backlight



- ●Low voltage driving (DC) is available without invertor
- ●No noise occurrence
- ●Luminous intensity: 50 cd/㎡
- ●Life : 20k hours

■ Pin Assignment

Pin no.	Symbol	Level	Function	
1	V_{SS}	—	Power supply	0V (GND)
2	V_{DD}	—		+5V
3	V_0	—		Contrast adjustment voltage
4	RS	H/L	Selects Registers H: Data register (when writing and reading) L: Instruction register (writing) Busy flag and address counter (reading)	
5	R/W	H/L	Read/write select signal H: Data read (Module→MPU) L: Data write (Module←MPU)	
6	E	H,H→L	Enable signal	
7~14	DB_0~DB_7	H/L	Data bus lines, see description below DB_4~DB_7 : High-order lines of data bus with three-state, bi-directional function for use in data transfer with the MPU. DB_7 may also be used to check the busy flag. DB_0~DB_3 : Low-order lines of data bus with three-state, bi-directional function for use in data transfer with the MPU. These lines are not used when interfacing with a 4-bit microprocessor.	

■ Electric Maximum Ratings

Item	Symbol	Min	Max	Unit	Remarks
Power supply for logic	V_{DD} - V_{SS}	-0.3	+7	V	
Signal input voltage	V_{IN}	$-0.3 \leqq V_{IN} \leqq V_{DD} + 0.3$		V	
Static electricity	—	—	100	V	See Note

Note : Electro-static discharge resistance is tested by charging a 200 pf capacitor and discharging it by contact with a interface connector pin.

FIG.1 Normal Temperature Type

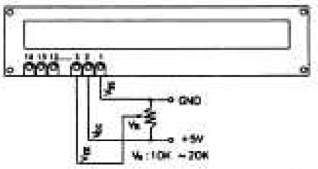
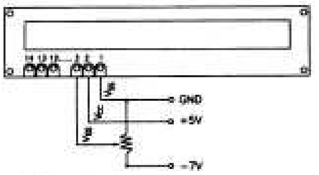


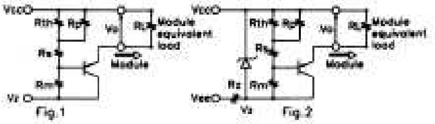
FIG.2 Extended Temperature Type



*Note:If V vary from recommended value, you cannot get proper contrast or viewing angle.

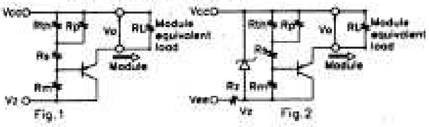
■ Examples of Temperature Compensation Circuits for Extended Temp Type. (Only for reference)

(A) 1/8Duty - 1/4Bias



Thermistor:Rth(25°C)=15[k-ohm],B=4200[K]
Resistors:Rp=30[k-ohm],Rs=6.8[k-ohm],Rm=3.3[k-ohm]
Transistors:PNP Type.
Vcc: +5V, Vss: 0V(Logic Suply)
Vz: -8V(-7.8~ -8.2V)
Vee<Vz,Rz=(Vz-Vee)/5 [k-ohm]

(B) 1/16Duty - 1/5Bias



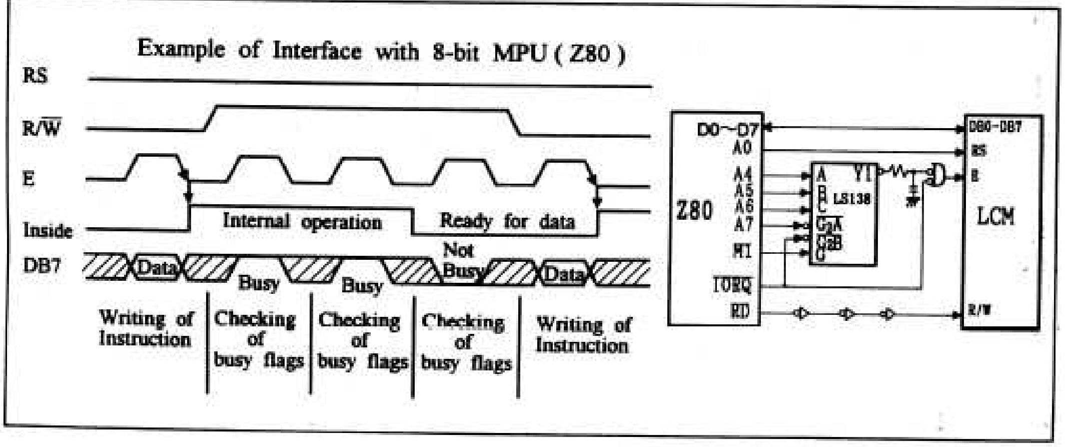
Thermistor:Rth(25°C)=15[k-ohm],B=4200[K]
Resistors:Rp=150[k-ohm],Rs=8.2[k-ohm],Rm=3.9[k-ohm]
Transistors:PNP Type.
Vcc: +5V, Vss: 0V(Logic Suply)
Vz: -11V(-10.725~ -11.275V)
Vee<Vz,Rz=(Vz-Vee)/5 [k-ohm]

■ Display Commands

Instruction	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description
1: Clear display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DD RAM address 0 in address counter
2: Return home	0	0	0	0	0	0	0	0	1	*	Sets DD RAM address 0 in address counter. Also returns display from being shifted to original position. DD RAM contents remain unchanged
3: Entry mode set	0	0	0	0	0	0	0	1	I/D	S	I/D=1: Increment I/D=0: Decrement S=1: Accompanies display shift
4: Display on/off	0	0	0	0	0	0	1	D	C	B	D=1/0: Display on/off C=1/0: Cursor on/off B=1: Blink of Cursor
5: Cursor/display shift	0	0	0	0	0	1	S/C	R/L	*	*	S/C=1: Display shift S/C=0: Cursor move R/L=0: Shift to left R/L=1: Shift to right
6: Function set	0	0	0	0	1	DL	N	F	*	*	DL=1: 8 bits, DL=0: 4 bits N=1: 2 lines, N=0: 1 line F=1: 5×10 dots, F=0: 5×8 dots
7: Set CG RAM address	0	0	0	1	ACG						ACG: CG RAM address
8: Set DD RAM address	0	0	1	ADD							ADD: DD RAM address corresponds to cursor address
9: Read busy flag/address counter	0	1	BF	AC							BF=1: Busy, BF=0: Not busy AC: Address counter used for both of CG and DD RAM address
10: Write data	1	0	Write data								Write data to CG or DD RAM
11: Read data	1	1	Read data								Read data from CG or DD RAM

☆ Execution Time (E_t) of Instruction: (Under condition of fcp or fosc = 270 KHz)
1 & 2 : E_t=1.52 ms
3~11 : E_t= 37 μs
☆ " * ": Either 0 or 1

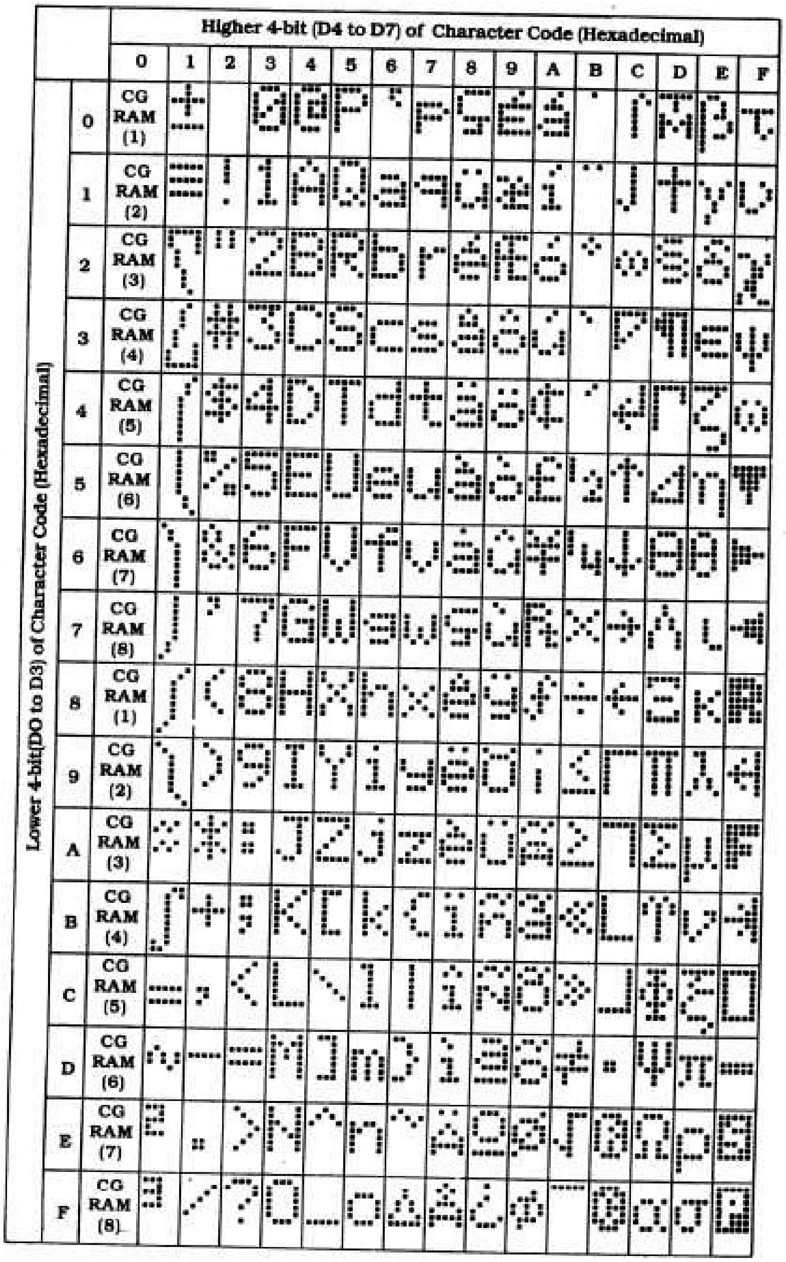
■ Connecting Block Diagram



Character Code Map (1)

		Higher 4-bit (D4 to D7) of Character Code (Hexadecimal)															
		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Lower 4-bit(DO to D3) of Character Code (Hexadecimal)	0	CG RAM (1)			0	@	P	`	p				ｰ	ﾀ	ﾐ	α	p
	1	CG RAM (2)		!	1	A	Q	a	q			｡	ｱ	ﾁ	ﾑ	ä	q
	2	CG RAM (3)		"	2	B	R	b	r			｢	ｲ	ﾂ	ﾒ	β	θ
	3	CG RAM (4)		#	3	C	S	c	s			｣	ｳ	ﾃ	ﾓ	ε	∞
	4	CG RAM (5)		$	4	D	T	d	t			､	ｴ	ﾄ	ﾔ	μ	Ω
	5	CG RAM (6)		%	5	E	U	e	u			･	ｵ	ﾅ	ﾕ	σ	ü
	6	CG RAM (7)		&	6	F	V	f	v			ｦ	ｶ	ﾆ	ﾖ	ρ	Σ
	7	CG RAM (8)		'	7	G	W	g	w			ｧ	ｷ	ﾇ	ﾗ	g	π
	8	CG RAM (1)		(	8	H	X	h	x			ｨ	ｸ	ﾈ	ﾘ	√	x̄
	9	CG RAM (2)		)	9	I	Y	i	y			ｩ	ｹ	ﾉ	ﾙ	⁻¹	y
	A	CG RAM (3)		*	:	J	Z	j	z			ｪ	ｺ	ﾊ	ﾚ	j	千
	B	CG RAM (4)		+	;	K	[	k	{			ｫ	ｻ	ﾋ	ﾛ	ˣ	万
	C	CG RAM (5)		,	<	L	¥	l	|			ｬ	ｼ	ﾌ	ﾜ	¢	円
	D	CG RAM (6)		-	=	M	]	m	}			ｭ	ｽ	ﾍ	ﾝ	£	÷
	E	CG RAM (7)		.	>	N	^	n	→			ｮ	ｾ	ﾎ	ﾞ	ñ	
	F	CG RAM (8)		/	?	O	_	o	←			ｯ	ｿ	ﾏ	ﾟ	ö	█

■ Character Code Map (2)

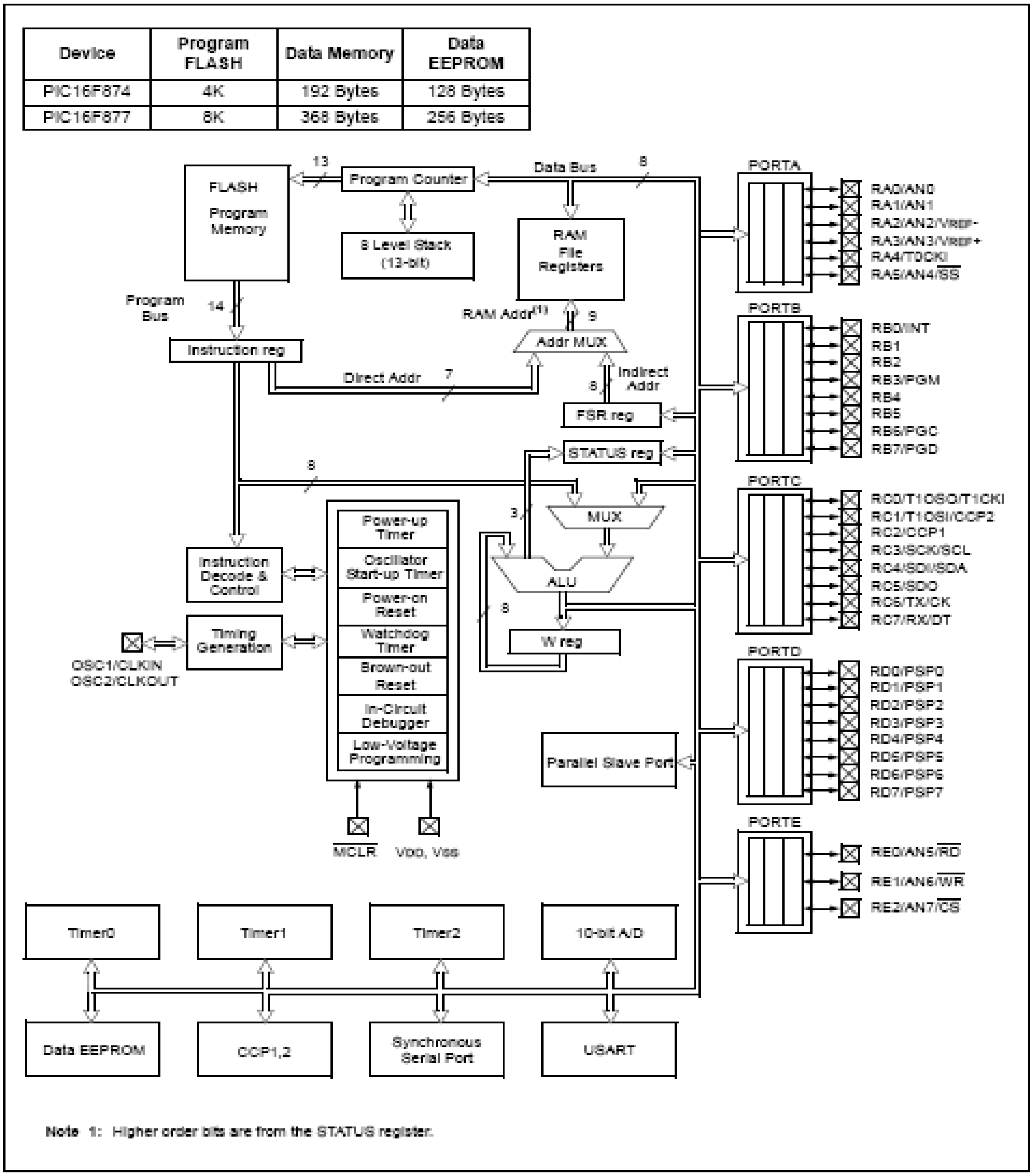

Higher 4-bit (D4 to D7) of Character Code (Hexadecimal)
0
1
2
3
4
5
6
7
8
9
A
B
C
D
E
F
Lower 4-bit(DO to D3) of Character Code (Hexadecimal)
0
CG RAM (1)
1
CG RAM (2)
2
CG RAM (3)
3
CG RAM (4)
4
CG RAM (5)
5
CG RAM (6)
6
CG RAM (7)
7
CG RAM (8)
8
CG RAM (1)
9
CG RAM (2)
A
CG RAM (3)
B
CG RAM (4)
C
CG RAM (5)
D
CG RAM (6)
E
CG RAM (7)
F
CG RAM (8)

APPENDIX C

PIC16F87X Data Sheet

PIC16F87X

FIGURE 1-2: PIC16F874 AND PIC16F877 BLOCK DIAGRAM



Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes

Note 1: Higher order bits are from the STATUS register.

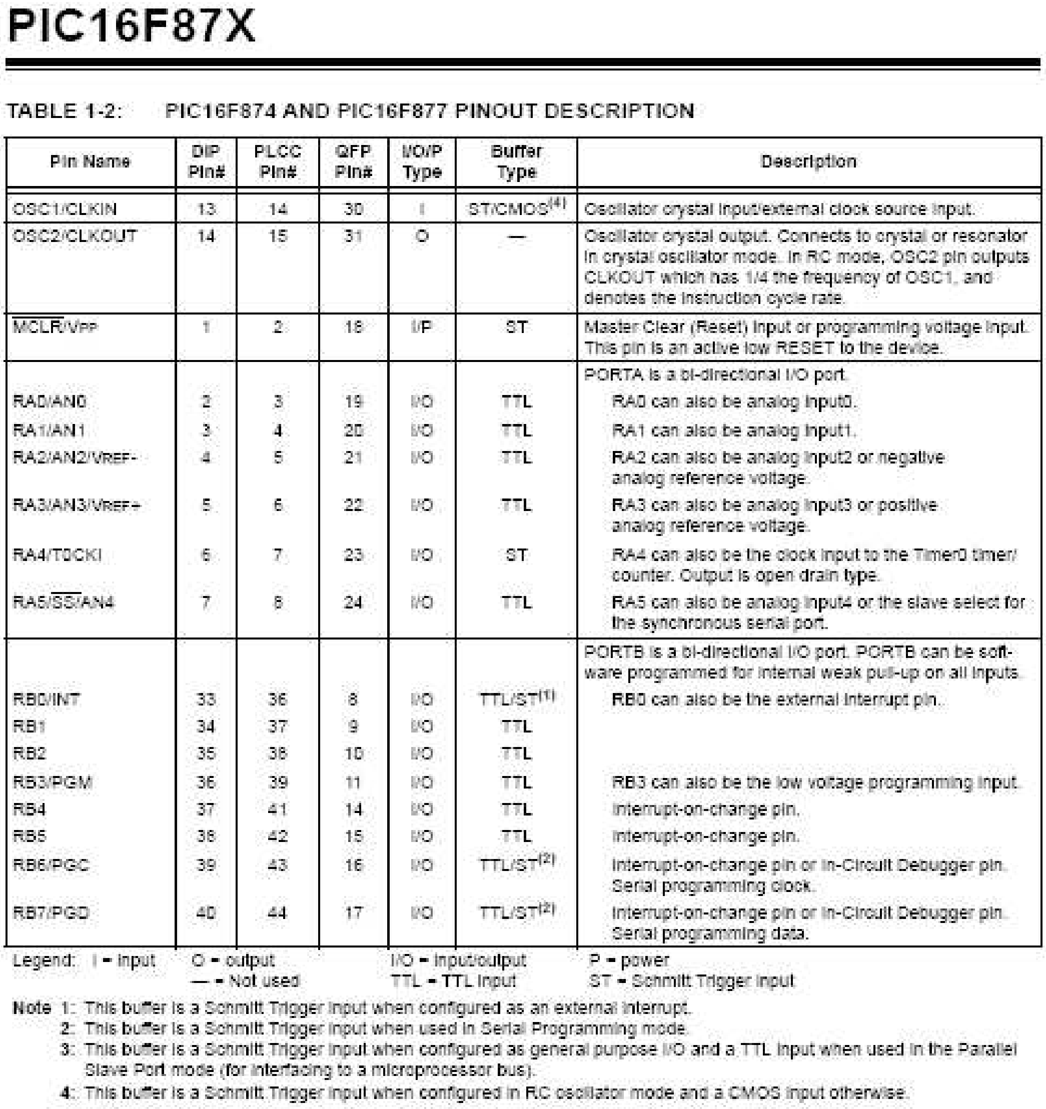


TABLE 1-2: PIC16F874 AND PIC16F877 PINOUT DESCRIPTION

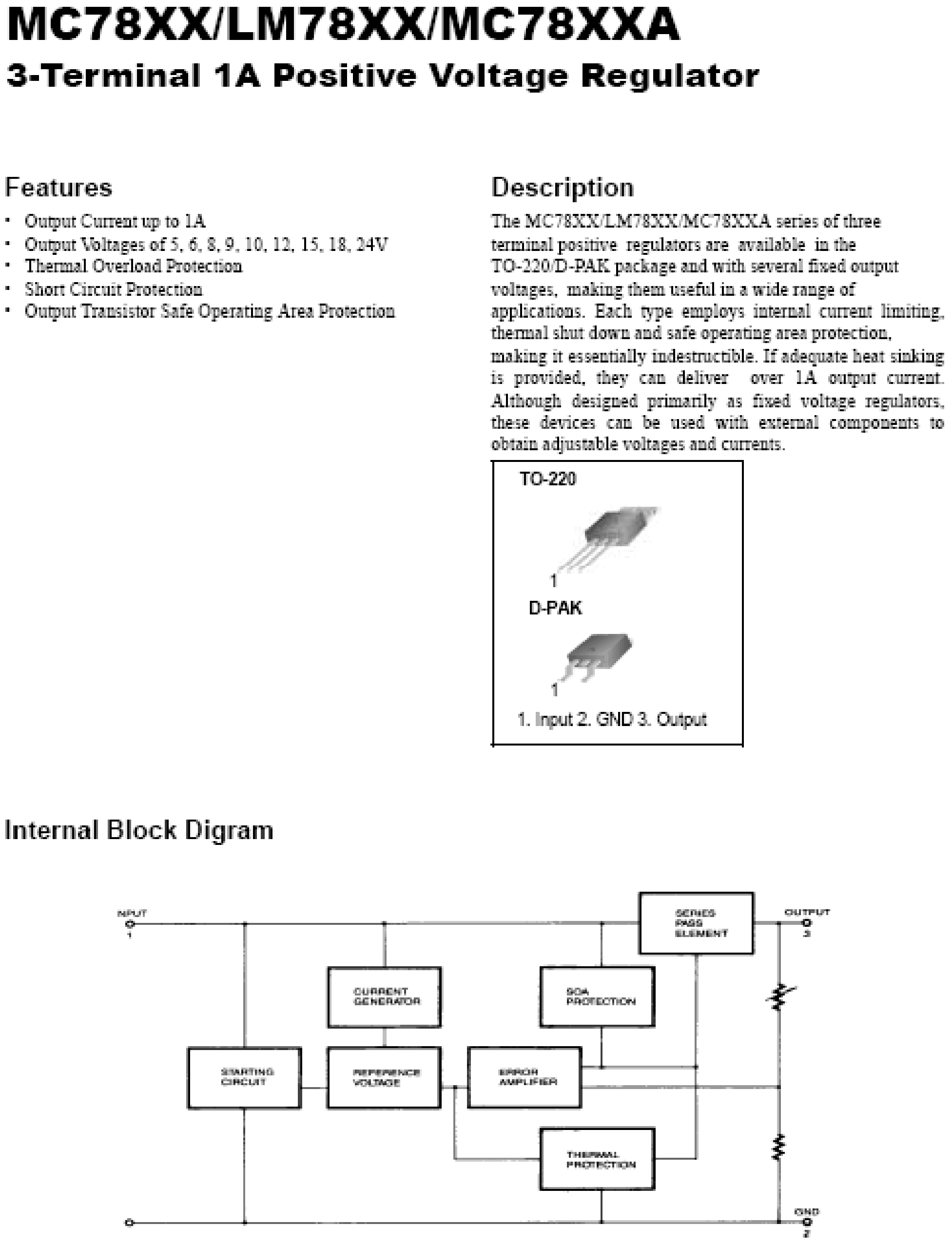
Pin Name	DIP Pin#	PLCC Pin#	QFP Pin#	I/O/P Type	Buffer Type	Description
OSC1/CLKIN	13	14	30	I	ST/CMOS$^{(4)}$	Oscillator crystal input/external clock source input.
OSC2/CLKOUT	14	15	31	O	—	Oscillator crystal output. Connects to crystal or resonator in crystal oscillator mode. In RC mode, OSC2 pin outputs 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.

APPENDIX D

LM7805 Data Sheet



MC78XX/LM78XX/MC78XXA

3-Terminal 1A Positive Voltage Regulator

Features

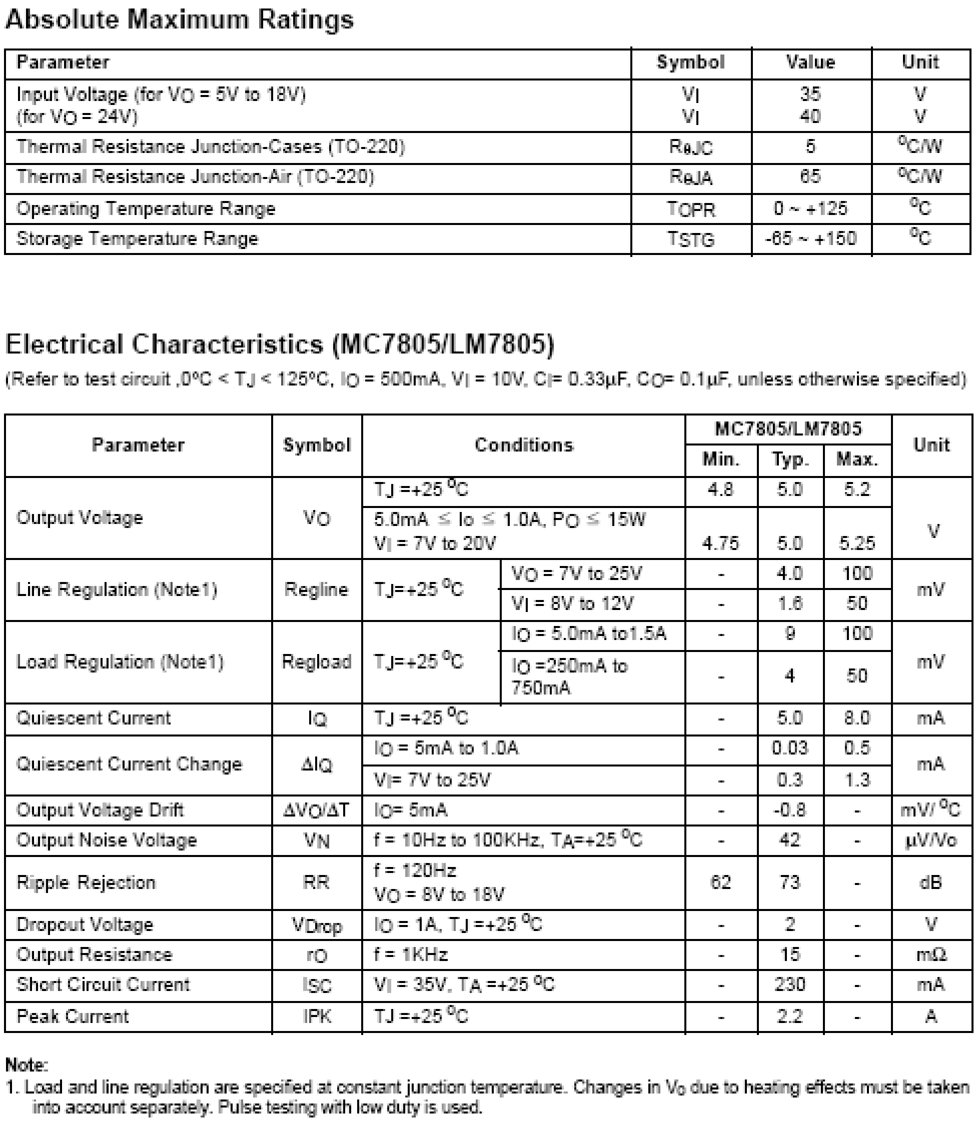
- Output Current up to 1A
- Output Voltages of 5, 6, 8, 9, 10, 12, 15, 18, 24V
- Thermal Overload Protection
- Short Circuit Protection
- Output Transistor Safe Operating Area Protection

Description

The MC78XX/LM78XX/MC78XXA series of three terminal positive regulators are available in the TO-220/D-PAK package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut down and safe operating area protection, making it essentially indestructible. If adequate heat sinking is provided, they can deliver over 1A output current. Although designed primarily as fixed voltage regulators, these devices can be used with external components to obtain adjustable voltages and currents.

1. Input 2. GND 3. Output

Internal Block Digram



Absolute Maximum Ratings

Parameter	Symbol	Value	Unit
Input Voltage (for V_O = 5V to 18V) (for V_O = 24V)	V_I V_I	35 40	V V
Thermal Resistance Junction-Cases (TO-220)	$R_{\theta JC}$	5	°C/W
Thermal Resistance Junction-Air (TO-220)	$R_{\theta JA}$	65	°C/W
Operating Temperature Range	T_{OPR}	0 ~ +125	°C
Storage Temperature Range	T_{STG}	-65 ~ +150	°C

Electrical Characteristics (MC7805/LM7805)

(Refer to test circuit ,0°C < T_J < 125°C, I_O = 500mA, V_I = 10V, C_I= 0.33µF, C_O= 0.1µF, unless otherwise specified)

Parameter	Symbol	Conditions		MC7805/LM7805			Unit
				Min.	Typ.	Max.	
Output Voltage	V_O	T_J =+25 °C		4.8	5.0	5.2	V
		5.0mA ≤ Io ≤ 1.0A, P_O ≤ 15W V_I = 7V to 20V		4.75	5.0	5.25	
Line Regulation (Note1)	Regline	T_J=+25 °C	V_O = 7V to 25V	-	4.0	100	mV
			V_I = 8V to 12V	-	1.6	50	
Load Regulation (Note1)	Regload	T_J=+25 °C	I_O = 5.0mA to1.5A	-	9	100	mV
			I_O =250mA to 750mA	-	4	50	
Quiescent Current	I_Q	T_J =+25 °C		-	5.0	8.0	mA
Quiescent Current Change	ΔI_Q	I_O = 5mA to 1.0A		-	0.03	0.5	mA
		V_I= 7V to 25V		-	0.3	1.3	
Output Voltage Drift	$\Delta V_O/\Delta T$	I_O= 5mA		-	-0.8	-	mV/ °C
Output Noise Voltage	V_N	f = 10Hz to 100KHz, T_A=+25 °C		-	42	-	µV/Vo
Ripple Rejection	RR	f = 120Hz V_O = 8V to 18V		62	73	-	dB
Dropout Voltage	V_{Drop}	I_O = 1A, T_J =+25 °C		-	2	-	V
Output Resistance	r_O	f = 1KHz		-	15	-	mΩ
Short Circuit Current	I_{SC}	V_I = 35V, T_A =+25 °C		-	230	-	mA
Peak Current	I_{PK}	T_J =+25 °C		-	2.2	-	A

Note:

1. Load and line regulation are specified at constant junction temperature. Changes in V_0 due to heating effects must be taken into account separately. Pulse testing with low duty is used.

APPENDIX E

Source Code

```
;
*********************************************
*******************************
;           File COINCTR5.ASM @ 4Mhz
            processor      16F877
            include        <P16F877.inc>
   __config   _XT_OSC & _WDT_OFF & _PWRTE_ON &
_LVP_OFF & _BODEN_OFF
;
*********************************************
*******************************
;            General Purpose RAM location: (STATUS-reg
RP1/RP0: x__x xxxx)
;                Bank_0: RP1/RP0 (00): 20H to 7FH (96
bytes)
;                Bank_1: RP1/RP0 (01): 20H to 6FH (80
bytes)
;                Bank_2: RP1/RP0 (10): 10H to 6FH (96
bytes)
;                Bank_3: RP1/RP0 (11): 10H to 6FH (96
bytes)
;            Note   : common access Bank_0 to Bank_3 :
70H to 7FH
;
*********************************************
*******************************
;                Variable Declaration
CounD05Hi     equu   H'20'        ;
CounD05Lo     equ   H'21'        ;
CounD10Hi     equ   H'22'        ;
CounD10Lo     equ   H'23'        ;
```

```
CounD25Hi     equ   H'24'        ;
CounD25Lo     equ   H'25'        ;
Count01Hi     equ   H'26'        ;
Count01Lo     equ   H'27'        ;
Count05Hi     equ   H'28'        ;
Count05Lo     equ   H'29'        ;
Count10Hi     equ   H'2A'        ;
Count10Lo     equ   H'2B'        ;
                                ;
AmtD05Hi      equ   H'30'        ;
AmtD05Lo      equ   H'31'        ;
AmtD05        equ   H'32'        ;
AmtD10Hi      equ   H'33'        ;
AmtD10Lo      equ   H'34'        ;
AmtD10        equ   H'35'        ;
AmtD25Hi      equ   H'36'        ;
AmtD25Lo      equ   H'37'        ;
AmtD25        equ   H'38'        ;
Amt01Hi       equ   H'39'        ;
Amt01Lo       equ   H'3A'        ;
Amt05Hi       equ   H'3B'        ;
Amt05Lo       equ   H'3C'        ;
Amt10Hi       equ   H'3D'        ;
Amt10Lo       equ   H'3E'        ;
                                ;
TotalDec      equ   H'40'        ;
TotalLo       equ   H'41'        ;
TotalHi       equ   H'42'        ;
AddDec        equ   H'43'        ;
AddLo         equ   H'44'        ;
AddHi         equ   H'45'        ;
```

```
TCountHi      equ   H'46'         ;
TCountLo      equ   H'47'         ;
                             ;
PortA_New     equ   H'50'         ;
PortA_Prev    equ   H'51'         ;
PortC_New     equ   H'52'         ;
PortC_Prev    equ   H'53'         ;
PortE_New     equ   H'54'         ;
PortE_Prev    equ   H'55'         ;
Disp_Sel      equ   H'56'         ;
                             ;
PortB_Buf     equ   H'70'         ;
Wait1_Val     equ   H'71'         ;
Wait2_Val     equ   H'72'         ;
Disp_Line     equ   H'73'         ;
Disp_Char     equ   H'74'         ;
Msg_Num       equ   H'75'         ;
                             ;
Temp1         equ   H'79'         ; temporary variable.
Temp2         equ   H'7A'         ;
Temp3         equ   H'7B'         ;
Temp4         equ   H'7C'         ;
W_TEMP         equ    H'7D'          ; temporary variable
for W.
STAT_TEMP      equ    H'7E'          ; temporary variable
for STATUS.
PCLATH_TEMP   equ   H'7F'         ;
;-----------------------------------------------------------------
--------
LCD_RAM_Buf   equ   H'20'         ; Bank 1
;
*********************************************
*******************************
;             Reset Vector Starts at Address 0x0000.
```

```
;
*********************************************
*******************************
            org   0x0000        ; start of reset vector.
            goto  Initialize    ;
                               ;
            org   0x0004        ; start of interrupt service
routine.
            goto  ISR_routine   ;
;
*********************************************
*******************************
;            Initialization Routine.
;
*********************************************
*******************************
Initialize:   clrf  TMR0          ; Clear TMR0
              clrf  INTCON        ; Disable Interrupts and
clear T0IF
            bcf   STATUS,RP1    ;
            bsf   STATUS,RP0    ; Select Bank 1
                               ;
                  movlw  B'11000100'     ; *  PortB  Pull-up
disabled.
                               ;   (useful if PortB is Input).
            movwf OPTION_REG    ;   prescaler of 1:32
                               ;
            movlw B'00000110'   ;Set PortA all Digital
            movwf ADCON1        ;
                               ;
            movlw B'11111111'   ;        0=OUT 1=IN
            movwf TRISA         ; Port A. 11xx xxxx:TTL
                               ;
            movlw B'00000000'   ;        0=OUT 1=IN
```

```
        movwf TRISB         ; Port B. xxxx xxxx:TTL
                    ;
        movlw B'11111111'   ;       0=OUT 1=IN
             movwf  TRISC           ;  Port  C.  xxxx
xxxx:schmitt
                    ;
        movlw B'00000000'   ;       0=OUT 1=IN
             movwf  TRISD           ;  Port  D.  xxxx
xxxx:schmitt
                    ;
        movlw B'00000111'   ;       0=OUT 1=IN
             movwf  TRISE           ;  Port  E.  0000
0xxx:schmitt
                    ;
        bcf   STATUS,RP0    ; Select Bank 0
                    ;
        call  Init_Var      ;
        call  Init_LCD      ;
        call  Disp_LCD      ;
                    ;
             bsf     INTCON,T0IE      ;  Enable  TMR0
Interrupt.
        bsf   INTCON,GIE    ; Enable All Interrupts.
;
********************************************
******************************
;       Main Program Starts Here.
;
********************************************
******************************
Main:     nop             ;
        goto  Main          ;
```

```
;
********************************************
******************************
;       The Interrupt Service Routine.
;
********************************************
******************************
ISR_routine:                ; Save Registers
        movwf W_TEMP        ;     W -> W_TEMP
        movf  STATUS,W      ; STATUS -> W
             movwf  STAT_TEMP         ;         W  ->
STAT_TEMP
        bcf   STATUS,RP0    ; Bank 0
                            ; Check which interrupt has
occurred.
        btfsc INTCON,T0IF   ; Timer0 Interrupt ?
        goto  TMR0int       ;
                            ; Other causes, disregard!
RestoreReg:                 ; Restore Registers
        movf  STAT_TEMP,W   ; STAT_TEMP -> W
        movwf STATUS        ;       W -> STATUS
        movf  W_TEMP,W      ;    W_TEMP -> W
                    ;
        retfie              ; Return from Interrupt.
;
********************************************
******************************
;       TIMER 0 (TMR0) Interrupt Service Routine.
;
********************************************
******************************
TMR0int:      bcf     INTCON,T0IF      ;  Reset  TMR0
Overflow Flag.
        movlw D'07'         ; store value to TMR0
        movwf TMR0          ;
                        ;
        call  Read_SW       ;
        call  Read_Sensor   ;
        call  Get_Total     ;
        call  Get_TCount    ;
        call  Disp_Dat0     ;
        call  Disp_Dat1     ;
        call  Disp_LCD      ;
                        ;
        goto  RestoreReg    ; done! Restore registers
& exit.
;
*********************************************
*******************************
Init_Var:   clrf   Msg_Num          ;
        movlw  D'0'             ;
        movwf  Msg_Num          ;
        call   Ld_Msg2RAM       ;
        clrf   PORTB            ;
        clrf   PORTD            ;
        movlw  D'1'             ;
        movwf  Disp_Sel         ;
                          ;
        movf   PORTA,W          ;
        movwf  PortA_New        ;
        movwf  PortA_Prev       ;
                          ;
        movf   PORTC,W          ;
        movwf  PortC_New        ;
        movwf  PortC_Prev       ;
                          ;
        movf   PORTE,W          ;
        movwf  PortE_New        ;
```

```
        movwf  PortE_Prev       ;
        call   Reset_Count      ;
                            ;
        return                  ;
                            ;
Reset_Count:  call   ResetD05       ;
        call   ResetD10         ;
        call   ResetD25         ;
        call   Reset01          ;
        call   Reset05          ;
        call   Reset10          ;
        clrf   TCountHi         ;
        clrf   TCountLo         ;
        return                  ;
                            ;
ResetD05:   clrf   CounD05Hi        ;
        clrf   CounD05Lo        ;
        clrf   AmtD05Hi         ;
        clrf   AmtD05Lo         ;
        clrf   AmtD05           ;
        return                  ;
                            ;
ResetD10:   clrf   CounD10Hi        ;
        clrf   CounD10Lo        ;
        clrf   AmtD10Hi         ;
        clrf   AmtD10Lo         ;
        clrf   AmtD10           ;
        return                  ;
                            ;
ResetD25:   clrf   CounD25Hi        ;
        clrf   CounD25Lo        ;
        clrf   AmtD25Hi         ;
        clrf   AmtD25Lo         ;
        clrf   AmtD25           ;
        return          ;
                        ;
Reset01:  clrf  Count01Hi     ;
        clrf  Count01Lo     ;
        clrf  Amt01Hi       ;
        clrf  Amt01Lo       ;
        return          ;
                        ;
Reset05:  clrf  Count05Hi     ;
        clrf  Count05Lo     ;
        clrf  Amt05Hi       ;
        clrf  Amt05Lo       ;
        return          ;
                        ;
Reset10:  clrf  Count10Hi     ;
        clrf  Count10Lo     ;
        clrf  Amt10Hi       ;
        clrf  Amt10Lo       ;
        return          ;
                        ;
;
********************************************
******************************
        org   0x100
Msg0:     addwf  PCL,F        ;
            ;01234567890123456789"
        dt    "P0.05x ___=   ___.00"
        dt    "P0.10x ___=   ___.00"
        dt    "P0.25x ___=   ___.00"
        dt    "TOTAL  ___: _,___.00"

            ;01234567890123456789"
Msg1:     dt    "P1.00x ___=   ___.00"
        dt    "P5.00x ___=   ___.00"
```

```
        dt    "P10.0x ___= _,___.00"
        dt    "TOTAL  ___: _,___.00"

;
********************************************
******************************
Read_Sensor:  movf   PORTA,W       ;Coin Sensor
        movwf  PortA_New      ;
                              ;
Rd_SWD05:   btfsc  PortA_New,0     ;
        goto   Rd_SWD05X      ;
        btfss  PortA_Prev,0    ;
        goto   Rd_SWD05X      ;
        call   INC_CD05       ;
        call   INC_AD05       ;
        movf   PortC_New,W     ;
        andlw  H'C0'         ;
        xorlw  H'C0'         ;
        btfsc  STATUS,Z       ;
        goto   Rd_SWD05X      ;
        movf   CounD05Hi,W     ;
        movwf  Temp1         ;
        movf   CounD05Lo,W     ;
        movwf  Temp2         ;
        call   Chk_Limit      ;
Rd_SWD05X:  nop                 ;
                              ;
Rd_SWD10:   btfsc  PortA_New,1     ;
        goto   Rd_SWD10X      ;
        btfss  PortA_Prev,1    ;
        goto   Rd_SWD10X      ;
        call   INC_CD10       ;
        call   INC_AD10       ;
        movf   PortC_New,W     ;
```

```
            andlw  H'C0'            ;
            xorlw  H'C0'            ;
            btfsc  STATUS,Z         ;
            goto   Rd_SWD10X        ;
            movf   CounD10Hi,W      ;
            movwf  Temp1            ;
            movf   CounD10Lo,W      ;
            movwf  Temp2            ;
            call   Chk_Limit        ;
Rd_SWD10X:  nop                     ;
                                    ;
Rd_SWD25:   btfsc  PortA_New,2      ;
            goto   Rd_SWD25X        ;
            btfss  PortA_Prev,2     ;
            goto   Rd_SWD25X        ;
            call   INC_CD25         ;
            call   INC_AD25         ;
            movf   PortC_New,W      ;
            andlw  H'C0'            ;
            xorlw  H'C0'            ;
            btfsc  STATUS,Z         ;
            goto   Rd_SWD25X        ;
            movf   CounD25Hi,W      ;
            movwf  Temp1            ;
            movf   CounD25Lo,W      ;
            movwf  Temp2            ;
            call   Chk_Limit        ;
Rd_SWD25X:  nop                     ;
                                    ;
Rd_SW01:    btfsc  PortA_New,3      ;
            goto   Rd_SW01X         ;
            btfss  PortA_Prev,3     ;
            goto   Rd_SW01X         ;
            call   INC_C01          ;
```

```
            call   INC_A01          ;
            movf   PortC_New,W      ;
            andlw  H'C0'            ;
            xorlw  H'C0'            ;
            btfsc  STATUS,Z         ;
            goto   Rd_SW01X         ;
            movf   Count01Hi,W      ;
            movwf  Temp1            ;
            movf   Count01Lo,W      ;
            movwf  Temp2            ;
            call   Chk_Limit        ;
Rd_SW01X:   nop                     ;
                                    ;
Rd_SW05:    btfsc  PortA_New,4      ;
            goto   Rd_SW05X         ;
            btfss  PortA_Prev,4     ;
            goto   Rd_SW05X         ;
            call   INC_C05          ;
            call   INC_A05          ;
            movf   PortC_New,W      ;
            andlw  H'C0'            ;
            xorlw  H'C0'            ;
            btfsc  STATUS,Z         ;
            goto   Rd_SW05X         ;
            movf   Count05Hi,W      ;
            movwf  Temp1            ;
            movf   Count05Lo,W      ;
            movwf  Temp2            ;
            call   Chk_Limit        ;
Rd_SW05X:   nop                     ;
                                    ;
Rd_SW10:    btfsc  PortA_New,5      ;
            goto   Rd_SW10X         ;
            btfss  PortA_Prev,5     ;
```

```
        goto   Rd_SW10X       ;
        call   INC_C10        ;
        call   INC_A10        ;
        movf   PortC_New,W    ;
        andlw  H'C0'          ;
        xorlw  H'C0'          ;
        btfsc  STATUS,Z       ;
        goto   Rd_SW10X       ;
        movf   Count10Hi,W    ;
        movwf  Temp1          ;
        movf   Count10Lo,W    ;
        movwf  Temp2          ;
        call   Chk_Limit      ;
Rd_SW10X:    nop              ;
                              ;
Rd_SensorX:  movf   PortA_New,W   ;
        movwf  PortA_Prev     ;
        return                ;
;
*********************************************
******************************
Read_SW:     movf   PORTC,W   ;
        movwf  PortC_New      ;
                              ;
Rd_SWStart:  btfsc  PortC_New,0   ;
        goto   Rd_SWStartX    ;
        btfss  PortC_Prev,0   ;
        goto   Rd_SWStartX    ;
        bsf    PORTD,0        ;
Rd_SWStartX: nop              ;
                              ;
Rd_SWStop:   btfsc  PortC_New,1   ;
        goto   Rd_SWStopX     ;
        btfss  PortC_Prev,1   ;
```

```
        goto   Rd_SWStopX     ;
        bcf    PORTD,0        ;
Rd_SWStopX:  nop              ;
                              ;
Rd_SWPage:   btfsc  PortC_New,2   ;
        goto   Rd_SWPageX     ;
        btfss  PortC_Prev,2   ;
        goto   Rd_SWPageX     ;
inc_page:    incf   Disp_Sel,F    ;
Rd_SWPageX:  nop              ;
                              ;
Rd_SWRst1:   btfsc  PortC_New,3   ;
        goto   Rd_SWRst1X     ;
        btfss  PortC_Prev,3   ;
        goto   Rd_SWRst1X     ;
        btfss  Disp_Sel,0     ;
        call   ResetD05       ;
        btfsc  Disp_Sel,0     ;
        call   Reset01        ;
Rd_SWRst1X:  nop              ;
                              ;
Rd_SWRst2:   btfsc  PortC_New,4   ;
        goto   Rd_SWRst2X     ;
        btfss  PortC_Prev,4   ;
        goto   Rd_SWRst2X     ;
        btfss  Disp_Sel,0     ;
        call   ResetD10       ;
        btfsc  Disp_Sel,0     ;
        call   Reset05        ;
Rd_SWRst2X:  nop              ;
                              ;
Rd_SWRst3:   btfsc  PortC_New,5   ;
        goto   Rd_SWRst3X     ;
        btfss  PortC_Prev,5   ;
        goto   Rd_SWRst3X      ;
        btfss  Disp_Sel,0      ;
        call   ResetD25        ;
        btfsc  Disp_Sel,0      ;
        call   Reset10         ;
Rd_SWRst3X:  nop               ;
                         ;
Read_SWX:  movf   PortC_New,W  ;
        movwf  PortC_Prev      ;
        return                 ;
;
********************************************
******************************
Chk_Limit:  clrf   Temp4       ;
                         ;
Chk_L50:   btfsc  PortC_New,6  ;
        goto   Chk_L50X        ;
        movf   Temp2,W         ;
        sublw  H'50'           ;
        btfss  STATUS,Z        ;
        goto   Chk_L50X        ;
        bcf    PORTD,0         ;
        bsf    Temp4,0         ;
Chk_L50X:  nop                 ;
                         ;
Chk_L100:  btfsc  PortC_New,7  ;
        goto   Chk_L100X       ;
        movf   Temp1,W         ;
        sublw  H'01'           ;
        btfss  STATUS,Z        ;
        goto   Chk_L100X       ;
        bcf    PORTD,0         ;
        bsf    Temp4,0         ;
Chk_L100X:  nop                ;
```

```
                               ;
Chk_LimitX:  return            ;
;
********************************************
******************************
INC_CD05:   incf   CounD05Lo,F    ;
        movf   CounD05Lo,W        ;
        movwf  Temp1              ;
        call   Dec_Adj            ;
        movf   Temp1,W            ;
        movwf  CounD05Lo          ;
        btfss  Temp3,0            ;
        goto   INC_CD05X          ;
        incf   CounD05Hi,F        ;
        movf   CounD05Hi,W        ;
        movwf  Temp1              ;
        call   Dec_Adj            ;
        movf   Temp1,W            ;
        movwf  CounD05Hi          ;
INC_CD05X:  return                ;
;
********************************************
******************************
INC_AD05:   movlw  H'05'          ;
        addwf  AmtD05,F           ;
        movf   AmtD05,W           ;
        movwf  Temp1              ;
        call   Dec_Adj            ;
        movf   Temp1,W            ;
        movwf  AmtD05             ;
                             ;
        btfss  Temp3,0            ;
        goto   INC_AD05X          ;
        incf   AmtD05Lo,F         ;
```

```
          movf   AmtD05Lo,W        ;
          movwf  Temp1             ;
          call   Dec_Adj           ;
          movf   Temp1,W           ;
          movwf  AmtD05Lo          ;
                                   ;
          btfss  Temp3,0           ;
          goto   INC_AD05X         ;
          incf   AmtD05Hi,F        ;
          movf   AmtD05Hi,W        ;
          movwf  Temp1             ;
          call   Dec_Adj           ;
          movf   Temp1,W           ;
          movwf  AmtD05Hi          ;
INC_AD05X:    return               ;
;
*********************************************
******************************
INC_CD10:    incf   CounD10Lo,F    ;
          movf   CounD10Lo,W       ;
          movwf  Temp1             ;
          call   Dec_Adj           ;
          movf   Temp1,W           ;
          movwf  CounD10Lo         ;
          btfss  Temp3,0           ;
          goto   INC_CD10X         ;
          incf   CounD10Hi,F       ;
          movf   CounD10Hi,W       ;
          movwf  Temp1             ;
          call   Dec_Adj           ;
          movf   Temp1,W           ;
          movwf  CounD10Hi         ;
INC_CD10X:    return               ;
```

```
;
*********************************************
******************************
INC_AD10:    movlw  H'10'          ;
          addwf  AmtD10,F          ;
          movf   AmtD10,W          ;
          movwf  Temp1             ;
          call   Dec_Adj           ;
          movf   Temp1,W           ;
          movwf  AmtD10            ;
                                   ;
          btfss  Temp3,0           ;
          goto   INC_AD10X         ;
          incf   AmtD10Lo,F        ;
          movf   AmtD10Lo,W        ;
          movwf  Temp1             ;
          call   Dec_Adj           ;
          movf   Temp1,W           ;
          movwf  AmtD10Lo          ;
                                   ;
          btfss  Temp3,0           ;
          goto   INC_AD10X         ;
          incf   AmtD10Hi,F        ;
          movf   AmtD10Hi,W        ;
          movwf  Temp1             ;
          call   Dec_Adj           ;
          movf   Temp1,W           ;
          movwf  AmtD10Hi          ;
INC_AD10X:    return               ;
;
*********************************************
******************************
INC_CD25:    incf   CounD25Lo,F    ;
          movf   CounD25Lo,W       ;
```

```
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
            movwf  CounD25Lo        ;
            btfss  Temp3,0          ;
            goto   INC_CD25X        ;
            incf   CounD25Hi,F      ;
            movf   CounD25Hi,W      ;
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
            movwf  CounD25Hi        ;
INC_CD25X:  return                  ;
;
**********************************************
*******************************
INC_AD25:   movlw  H'25'            ;
            addwf  AmtD25,F         ;
            movf   AmtD25,W         ;
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
            movwf  AmtD25           ;
                                ;
            btfss  Temp3,0          ;
            goto   INC_AD25X        ;
            incf   AmtD25Lo,F       ;
            movf   AmtD25Lo,W       ;
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
            movwf  AmtD25Lo         ;
                                ;
            btfss  Temp3,0          ;
```

```
            goto   INC_AD25X        ;
            incf   AmtD25Hi,F       ;
            movf   AmtD25Hi,W       ;
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
            movwf  AmtD25Hi         ;
INC_AD25X:  return                  ;
;
**********************************************
*******************************
INC_C01:    incf   Count01Lo,F      ;
            movf   Count01Lo,W      ;
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
            movwf  Count01Lo        ;
            btfss  Temp3,0          ;
            goto   INC_C01X         ;
            incf   Count01Hi,F      ;
            movf   Count01Hi,W      ;
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
            movwf  Count01Hi        ;
INC_C01X:   return                  ;
;
**********************************************
*******************************
INC_A01:    incf   Amt01Lo,F        ;
            movf   Amt01Lo,W        ;
            movwf  Temp1            ;
            call   Dec_Adj          ;
            movf   Temp1,W          ;
```

```
        movwf  Amt01Lo          ;
        btfss  Temp3,0          ;
        goto   INC_A01X         ;
        incf   Amt01Hi,F        ;
        movf   Amt01Hi,W        ;
        movwf  Temp1            ;
        call   Dec_Adj          ;
        movf   Temp1,W          ;
        movwf  Amt01Hi          ;
INC_A01X:   return              ;
;
*******************************************
*****************************
INC_C05:    incf   Count05Lo,F      ;
        movf   Count05Lo,W      ;
        movwf  Temp1            ;
        call   Dec_Adj          ;
        movf   Temp1,W          ;
        movwf  Count05Lo        ;
        btfss  Temp3,0          ;
        goto   INC_C05X         ;
        incf   Count05Hi,F      ;
        movf   Count05Hi,W      ;
        movwf  Temp1            ;
        call   Dec_Adj          ;
        movf   Temp1,W          ;
        movwf  Count05Hi        ;
INC_C05X:   return              ;
;
*******************************************
*****************************
INC_A05:    movlw  D'5'             ;
        addwf  Amt05Lo,F        ;
        movf   Amt05Lo,W        ;
        movwf  Temp1            ;
        call   Dec_Adj          ;
        movf   Temp1,W          ;
        movwf  Amt05Lo          ;
        btfss  Temp3,0          ;
        goto   INC_A05X         ;
        incf   Amt05Hi,F        ;
        movf   Amt05Hi,W        ;
        movwf  Temp1            ;
        call   Dec_Adj          ;
        movf   Temp1,W          ;
        movwf  Amt05Hi          ;
INC_A05X:   return              ;
;
*******************************************
*****************************
INC_C10:    incf   Count10Lo,F      ;
        movf   Count10Lo,W      ;
        movwf  Temp1            ;
        call   Dec_Adj          ;
        movf   Temp1,W          ;
        movwf  Count10Lo        ;
        btfss  Temp3,0          ;
        goto   INC_C10X         ;
        incf   Count10Hi,F      ;
        movf   Count10Hi,W      ;
        movwf  Temp1            ;
        call   Dec_Adj          ;
        movf   Temp1,W          ;
        movwf  Count10Hi        ;
INC_C10X:   return              ;
;
*******************************************
*****************************
```

```
INC_A10:    movlw  D'10'          ;
            addwf  Amt10Lo,F      ;
            movf   Amt10Lo,W      ;
            movwf  Temp1          ;
            call   Dec_Adj        ;
            movf   Temp1,W        ;
            movwf  Amt10Lo        ;
            btfss  Temp3,0        ;
            goto   INC_A10X       ;
            incf   Amt10Hi,F      ;
            movf   Amt10Hi,W      ;
            movwf  Temp1          ;
            call   Dec_Adj        ;
            movf   Temp1,W        ;
            movwf  Amt10Hi        ;
INC_A10X:   return                ;
;
*********************************************
*****************************
Get_Total:  clrf   TotalDec       ;
            clrf   TotalLo        ;
            clrf   TotalHi        ;
                                  ;
            movf   AmtD05,W       ;
            movwf  AddDec         ;
            movf   AmtD05Lo,W     ;
            movwf  AddLo          ;
            movf   AmtD05Hi,W     ;
            movwf  AddHi          ;
            call   Do_Add         ;
                                  ;
            movf   AmtD10,W       ;
            movwf  AddDec         ;
            movf   AmtD10Lo,W     ;
```

```
            movwf  AddLo          ;
            movf   AmtD10Hi,W     ;
            movwf  AddHi          ;
            call   Do_Add         ;
                                  ;
            movf   AmtD25,W       ;
            movwf  AddDec         ;
            movf   AmtD25Lo,W     ;
            movwf  AddLo          ;
            movf   AmtD25Hi,W     ;
            movwf  AddHi          ;
            call   Do_Add         ;
                                  ;
            clrf   AddDec         ;
            movf   Amt01Lo,W      ;
            movwf  AddLo          ;
            movf   Amt01Hi,W      ;
            movwf  AddHi          ;
            call   Do_Add         ;
                                  ;
            clrf   AddDec         ;
            movf   Amt05Lo,W      ;
            movwf  AddLo          ;
            movf   Amt05Hi,W      ;
            movwf  AddHi          ;
            call   Do_Add         ;
                                  ;
            clrf   AddDec         ;
            movf   Amt10Lo,W      ;
            movwf  AddLo          ;
            movf   Amt10Hi,W      ;
            movwf  AddHi          ;
            call   Do_Add         ;
                                  ;
```

```
Get_TotalX:   return                ;
;
********************************************
******************************
Get_TCount:   movf   CounD05Lo,W       ;
        movwf  TCountLo          ;
        movf   CounD05Hi,W       ;
        movwf  TCountHi          ;
                                 ;
        movf   CounD10Lo,W       ;
        movwf  Temp2             ;
        movf   TCountLo,W        ;
        movwf  Temp1             ;
        call   Add_BCD           ;
        movf   Temp1,W           ;
        movwf  TCountLo          ;
                                 ;
        btfsc  Temp3,0           ;
        incf   TCountHi,F        ;
        movf   CounD10Hi,W       ;
        movwf  Temp2             ;
        movf   TCountHi,W        ;
        movwf  Temp1             ;
        call   Add_BCD           ;
        movf   Temp1,W           ;
        movwf  TCountHi          ;
                                 ;
        movf   CounD25Lo,W       ;
        movwf  Temp2             ;
        movf   TCountLo,W        ;
        movwf  Temp1             ;
        call   Add_BCD           ;
        movf   Temp1,W           ;
        movwf  TCountLo          ;
                                 ;
        btfsc  Temp3,0           ;
        incf   TCountHi,F        ;
        movf   CounD25Hi,W       ;
        movwf  Temp2             ;
        movf   TCountHi,W        ;
        movwf  Temp1             ;
        call   Add_BCD           ;
        movf   Temp1,W           ;
        movwf  TCountHi          ;
                                 ;
        movf   Count01Lo,W       ;
        movwf  Temp2             ;
        movf   TCountLo,W        ;
        movwf  Temp1             ;
        call   Add_BCD           ;
        movf   Temp1,W           ;
        movwf  TCountLo          ;
                                 ;
        btfsc  Temp3,0           ;
        incf   TCountHi,F        ;
        movf   Count01Hi,W       ;
        movwf  Temp2             ;
        movf   TCountHi,W        ;
        movwf  Temp1             ;
        call   Add_BCD           ;
        movf   Temp1,W           ;
        movwf  TCountHi          ;
                                 ;
        movf   Count05Lo,W       ;
        movwf  Temp2             ;
        movf   TCountLo,W        ;
        movwf  Temp1             ;
        call   Add_BCD           ;
        movf   Temp1,W          ;
        movwf  TCountLo         ;
                       ;
        btfsc  Temp3,0          ;
        incf   TCountHi,F       ;
        movf   Count05Hi,W      ;
        movwf  Temp2            ;
        movf   TCountHi,W       ;
        movwf  Temp1            ;
        call   Add_BCD          ;
        movf   Temp1,W          ;
        movwf  TCountHi         ;
                       ;
        movf   Count10Lo,W      ;
        movwf  Temp2            ;
        movf   TCountLo,W       ;
        movwf  Temp1            ;
        call   Add_BCD          ;
        movf   Temp1,W          ;
        movwf  TCountLo         ;
                       ;
        btfsc  Temp3,0          ;
        incf   TCountHi,F       ;
        movf   Count10Hi,W      ;
        movwf  Temp2            ;
        movf   TCountHi,W       ;
        movwf  Temp1            ;
        call   Add_BCD          ;
        movf   Temp1,W          ;
        movwf  TCountHi         ;
                       ;
Get_TCountX:  return            ;
```

```
;
*********************************************
******************************
Add_BCD:     clrf   Temp3           ;
Add_BCDLo:   movf   Temp2,W         ;
             andlw  H'0F'           ;
             addwf  Temp1,F         ;
             btfsc  STATUS,DC       ;
             goto   Adj_BCDLo       ;
             movf   Temp1,W         ;
             andlw  H'0F'           ;
             sublw  D'9'            ;
             btfsc  STATUS,C        ;
             goto   Add_BCDHi       ;
Adj_BCDLo:   movlw  D'6'            ;
             addwf  Temp1,F         ;
Add_BCDHi:   movf   Temp2,W         ;
             andlw  H'F0'           ;
             addwf  Temp1,F         ;
             btfsc  STATUS,C        ;
             goto   Adj_BCDHi       ;
             swapf  Temp1,W         ;
             andlw  H'0F'           ;
             sublw  D'9'            ;
             btfsc  STATUS,C        ;
             goto   Add_BCDX        ;
Adj_BCDHi:   movlw  H'60'           ;
             addwf  Temp1,F         ;
             bsf    Temp3,0         ;
Add_BCDX:    return                 ;
;
*********************************************
******************************
Do_Add:      movf   AddDec,W        ;
```

```
        movwf  Temp2            ;
        movf   TotalDec,W       ;
        movwf  Temp1            ;
        call   Add_BCD          ;
        movf   Temp1,W          ;
        movwf  TotalDec         ;
                                ;
        btfsc  Temp3,0          ;
        incf   TotalLo,F        ;
        movf   AddLo,W          ;
        movwf  Temp2            ;
        movf   TotalLo,W        ;
        movwf  Temp1            ;
        call   Add_BCD          ;
        movf   Temp1,W          ;
        movwf  TotalLo          ;
                                ;
        btfsc  Temp3,0          ;
        incf   TotalHi,F        ;
        movf   AddHi,W          ;
        movwf  Temp2            ;
        movf   TotalHi,W        ;
        movwf  Temp1            ;
        call   Add_BCD          ;
        movf   Temp1,W          ;
        movwf  TotalHi          ;
                                ;
Do_AddX:    return              ;
;
*******************************************************
*************************************
```

```
Dec_Adj:    clrf   Temp3        ;
        movf   Temp1,W          ;
        andlw  H'0F'            ;
        sublw  D'9'             ;
        btfsc  STATUS,C         ;
        goto   Dec_Adj10        ;
        movlw  D'6'             ;
        addwf  Temp1,F          ;
Dec_Adj10:  swapf  Temp1,W      ;
        andlw  H'0F'            ;
        sublw  D'9'             ;
        btfsc  STATUS,C         ;
        goto   Dec_AdjX         ;
        bsf    Temp3,0          ;
        movlw  H'60'            ;
        addwf  Temp1,F          ;
Dec_AdjX:   return              ;
;
*******************************************************
*************************************
        include   <Coin5LCD.inc>
;
*******************************************************
**************************************
        end                     ;
;
*******************************************************
**************************************
```