UF UNIVERSITY of FLORIDA
The Foundation for The Gator Nation

EEL 4914 Senior Design
Final Design Report

Fall 2006

Automatic Pill Dispenser

Team: Not a Clue

Submitted by:

B. Bowers – bowersb@ufl.edu – 321.987.7277
M. Preston – mitchelp@ufl.edu – 352.372.5980

I. Project Assumptions and Objectives

Background Information

As the cost of in-home medical care rises, it has become more and more incumbent upon individuals in need of supervised medical care to find a means in which to lower their medical care costs. As such, many individuals who require the administering of many dosages of medications at specific times have turned to devices such as automatic pill dispensers to alleviate the need for an in home nurse on a daily basis. These dispensers range in cost from $200 up to $800. These dispensers allow for an in home medical care provider to have a means to regulate a patient's medications without having to constantly supervise the patient. Typical features on these dispensers include automatic pill dispensing at regular intervals, audible warnings, as well as a connection to either a phone line or the internet for monitoring purposes by the medical care provider.

As previously mentioned, many of these devices are rather expensive and can be somewhat cumbersome. Through the use of a simple microprocessor and motor unit an automatic pill dispenser can be produced for a much cheaper price and be much more user friendly.

Project Definition

The team will design and build an automatic pill dispenser. The product will consist of a circular base with 22 fan-like blades that rotate about the central axis. The blades will form the compartments where pills can be manually placed for dispensing at predetermined times. The dispenser will be controlled by a microprocessor that interfaces with an LED display, as well as an alphanumeric keypad that will be utilized as a source for the inputting of data, and selecting from preprogrammed menu items. The user will be able to input the time(s) of day that pills will be dispensed, as well as any warnings and/or precautions that must be followed when the patient takes his or her pills. Lastly, the dispenser will visually and audibly notify the patient when pills are being dispensed, and will also store the time of day that the patient took his or her medications. Finally, the dispenser will automatically adjust the time of the next medication dispersal if necessary, to avoid dosages of medication being taken too closely together. The project will be realized with the development of:

i. A compact, pill-dispensing unit that can be placed on a table or countertop.
ii. A microprocessor unit that will control the form and function of the device according to specifications.

Scope of Project

The pill dispenser will be developed with off-the-shelf technology for the design and implementation of the project. The end goal is not to develop any new technologies associated with current manufactured dispensers. Rather, the goal is to design a unit with the same basic functionality, but for a much cheaper price.

Major Objectives

- **i.** Construct a device that is relatively small and lightweight.
- **ii.** Develop the software in such a way that patients receive their medication reliably and safely as prescribed by their physician.
- **iii.** Use as much off the shelf technology, as well as harvest parts from other systems to keep costs low.
- **iv.** Develop a device that can perform all the necessary functions as stated in the project abstract.

Product Expectations

- **i.** An audio and visual alarm to notify patient that medication has been dispensed.
- **ii.** A microprocessor controlled system that will automatically dispense medications at the preset time of day.
- **iii.** Software that will monitor and record the time that medications have been taken by the patient.
- **iv.** Software that will automatically adjust future medication dispensation based on when the medication was taken by the patient.
- **v.** A mechanical locking mechanism that will keep the patient from over medicating.
- **vi.** An LED display that will give pertinent instructions about the medication to the patient.

II. Customer Requirements

Performance

- Lightweight
- Easy to use for both caretaker and patient
- Well constructed to avoid potential tampering
- Bright warning LED
- 70+ decibel audio warning

Serviceability

- Long life span
- Easy to repair

Features

- Lightweight
- Small dimensions
- Ability to reset the system
- Locking mechanism
- Processor controlled automation for pill dispensing
- Audio and visual alarm
- LED display for patient notification/instructions
- Ability to store times that patient received medications

Reliability

- Stable software
- Mechanical devices encased for safety and durability

Cost

- Low cost
- Off the shelf components that are easily replaced

Safety

- Electrical components encased
- Mechanical parts and motor encased

III. Analysis of Competitive Products

After researching competitive products via the internet, the team made a decision as to what features are necessary to the construction of a useful product. Furthermore, certain features were added that were not typical to current market products. We also assessed the viability of some more complicated features given the amount of time we have to develop out product.

Typical Features of Market Products

- Fully Automatic Pill Dispenser
- Easy Set-up
- Simple to use
- Unlimited # dispenses per day (up to 28 times per day)
- Medication trays
- Lockable with key
- Long Alarm time duration (up to 60 minutes)

Additional Features

There was only one feature in particular that the team felt may not be feasible within the scope of the time given to accomplish the project. Some of the higher end dispensers contain a feature that will notify the patient's caretaker by either phone, e-mail, or both that the patient did not receive their medication within a timely manner. Though this feature is not impossible, it would be quite difficult to reproduce within the time given for design and production of the product.

IV. Concept Selection

In this section, current market product features were taken into account to decide on the features we would include in our pill dispenser. Having taken aforementioned features and concerns into account we decided on the following general concept:

The Programmable Medication Dispenser (PMD) design allows the caregiver to reliably administer medications to a patient without needing to be present every time the medication is scheduled. The PMD allows the caregiver to preprogram up to 21 medication doses through an ergonomically designed interface, utilizing an alphanumeric keypad, and LED display. The basic concept for out project is listed below with a final concept shown in Figure 1.

Hardware Concept Design

Power Supply: The power supply design will provide the necessary power requirements of the PMD. The design requirements are 5 VDC for the microprocessor, and motor controller; additionally, 12 and/or 15 VDC maybe necessary for the motor controller. The current requirements will be mainly dictated by the motor controller design; while the current requirements for the microprocessor are in the mA range, the motor controller may require several amps. Finally, the power supply may require battery backup to avoid loss of user input selections and time keeping functions; depending on the microprocessor and memory designs selected.

Keypad: The keypad input will be a standard 16 key alphanumeric keypad.

Motor: The motor for the PMD will be a stepper motor. The stepper motor selection will be determined by the torque requirements of the dispenser and the number of step divisions required for reliable and accurate medication delivery.

Pill Container and Dispenser: The pill container design will have 22 slots with 21 available for medications. The 21 slots will be labeled so that the caregiver can ensure they are setting the alarm for the correct slot. Additionally, the pill container will incorporate interlock sensors to protect the caregiver and user from harm. Finally, the sensors will provide input to the microprocessor for determining access doors status, medication slot positioning and time feedback of the patient accessing the medications.

Software Concept Design

Microprocessor: The microprocessor will be selected to meet required functionality of the PMD, without wasting money on unneeded features. Additionally, the microprocessor will be chosen so that external memory can be added if necessary, depending on the final algorithm design.

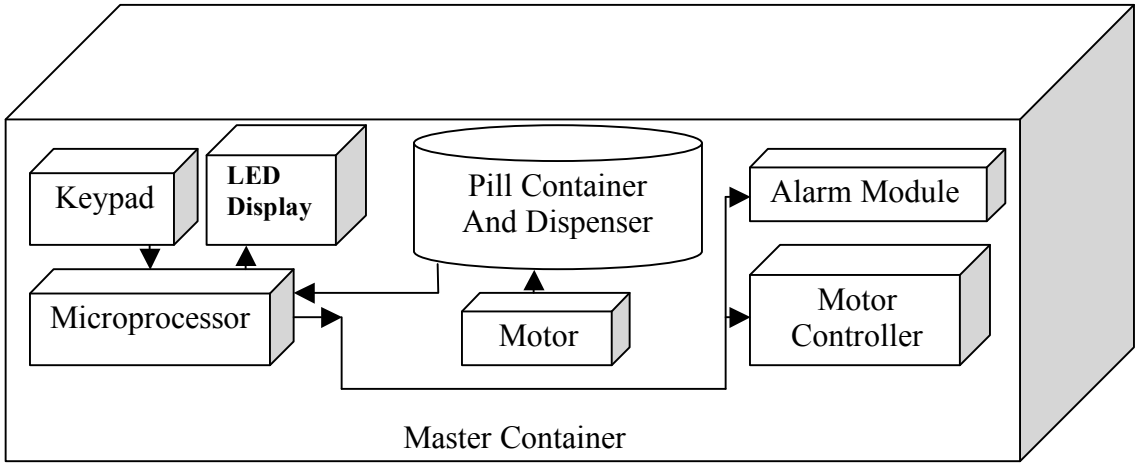
Alarm Module: The alarm module will provide an audible alarm tone. Additionally, the alarm module will provide a visual cue, through the use of LEDs. The design of the alarm module will be to provide a limited alarm function in case of loss of main AC power.

Motor Controller: The design of the motor controller will be determined by the current requirements of the stepper motor selected. The motor controller will take logic inputs from the microprocessor and supply enough current to the stepper motor to meet maximum torque requirements.

LED: The LED display will be a simple black and white, multi-line line display. The LED display will provide information to the caregiver for the purposes of programming and will give feedback on medication compliance by the patient. Additionally, the LED display will convey pre-selected precautions to the patient concerning the medications being currently dispensed.

Complete Design Concept

Programmable Medication Dispenser Block Diagram



V. Project Plans and Scheduling

Our team developed a time line, as well as delegated responsibilities in such a way that all team members must participate in an equal manner. Though we decided to work together and consult as a team on all aspects of the design, certain members were responsible for specific aspects of the project as detailed below.

5.1 **Parts** / Mitchel Preston

All parts were ordered by no later than 9/11/06, and the last part was acquired by 9/20/06.

5.2 **Circuit Design** / Benjamin Bowers

This part of the project was simply comprised of designing a theoretical circuit utilizing software such as PSPICE to test potential circuit implementations for our project. Before anything was soldered, we needed to know that we have a circuit design that will meet the specifications. Furthermore, this aspect of the project helped determine any and all possible challenges met throughout the course of the project given the parts that we decided to utilize.

5.3 **Circuit Implementation** / Benjamin Bowers

This part of the project was the physical implementation of our project once we were satisfied with the results. During this phase of the project, breadboard circuit testing was conducted in order to determine whether our project meets the specifications as well as what was predicted by the software programs. Furthermore, the physical soldering and packaging of the circuit will also took place in this phase of the design once breadboard testing was complete.

5.4 **Product Construction** / Benjamin Bowers (primary) & Mitchel Preston

This phase of the project consisted of the physical building of the unit, itself. Though one member is listed as being primarily responsible, both team members actively participated in the build.

5.5 **Software Implementation** / Mitchel Preston

This phase of the project consisted of the design of the software that controlled the function of the motor, alarm circuit, and memory/LED readout.

Project Timeline (Gantt Chart)

	Wk 1 8/21/06	Wk 2 8/28/06	Wk 3 9/4/06	Wk 4 9/11/06	SpBk 9/18/06	Wk 6 9/19/06	Wk 7 9/20/06	Wk8 9/21/06	Wk9 9/22/06	Wk10 9/23/06	Wk11 9/24/06
Parts /Mitchel P.	X	X	X	X							
Circuit Design/ Ben B.	X	X	X	X							
Circuit Implementation /Ben B.				X	X	X					
Product Construction/ Ben B. (primary) Mitchel P.						X	X	X	X	X	
Software Implementation/ Mitchel P.							X	X	X	X	

VI. Unit Cost

One of the goals of this design was to produce a product comparable to current market pill dispensers but at a more affordable cost. The lowest cost unit that was found through internet research was approximately $150. Though cheaper units can be bought, the aforementioned unit was the cheapest one that the team could locate with features similar to our design. The cost for the team's design can be seen in the chart below. However, it must be noted that many of the components were provided by Mike Stapleton, free of charge.

Design Unit Cost Analysis Table

Item	Cost per unit
Current Limiting Diodes	Free from Mike
Stepping Motor	$10.00
Resistors	$3.00
L293 Motor Driver	$1.00
Plastiboard	$5.00
3/4 inch Dowel Rod	$0.50
555 Timer	$1.69
PNP transistors	$2.59
Bright Red LED	$2.69
75 dB Buzzer	$3.29
Capacitors	Free from Mike
4 MHz Crystal	Free from Mike
D1307 Chip	Free from Mike
32 Crystal	Free from Mike
Lithium Battery	Free from Mike
Battery Holder	Free from Mike
PIC16F877A I/P	$7.50
Switching Diode N4914	Free from Mike
Cookie Tin	$2.00
Screws	$6.00
LCD	Free from Mike
16 Key Alphanumeric Keypad	MicroP
Total Unit Cost:	**$45.26**

The total unit cost was well under $150. However, this number does not take into account the cost of many minor items such as batteries, capacitors, and crystal chips. Even so, the total unit cost would still be well under the original goal. As such, the team's design is an extremely affordable unit.

VII. Completed Design Analysis

7.1 Design Changes

There were very few changes to the original design. Originally, our team wanted to be able to program the module for all 21 slots. However, it was later discovered that the PIC16F877 I/P chip contained only 8kB of memory. As such, there was not enough memory to allow for this feature after all of the other software design features were implemented. Therefore, the team decided to allow for the ability to program two slot times in order to show proof of concept understanding and implementation. The chart below shows lists all of the original design features with a verification mark next to those that were met.

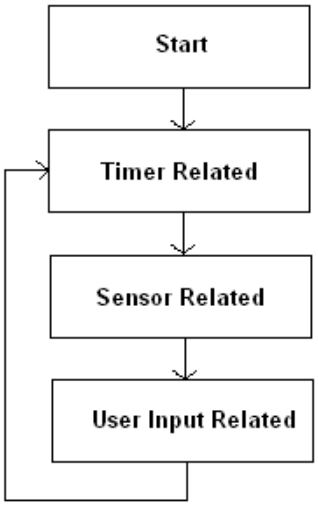
Feature	Functional	Not Functional
Ability to Keep Time	X	
Audiovisual Alarm	X	
Ability to Set Multiple Alarm Times	X	
View Past Alarms	X	
View Current Alarm Times	X	
Auto Positioning of the Carousel	X	
Time Battery Backup	X	
Backup Alarm Memory	X	
Backup Alarm Adjust	X	
Keypad Programmability	X	
LCD Functional	X	

7.2 Design Problems/Bugs

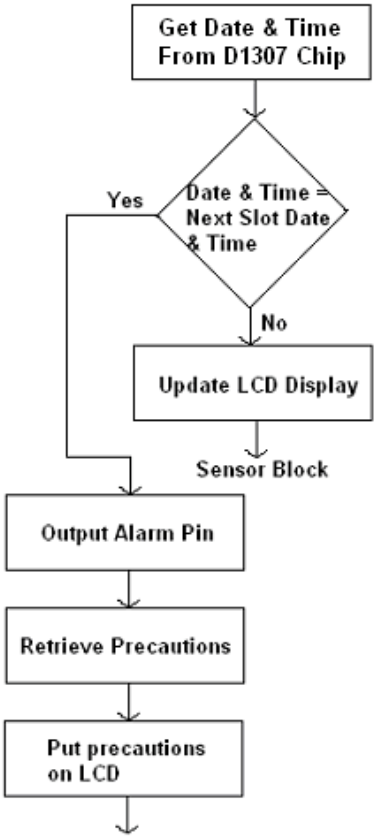
Despite taking more time than was expected, we ran into very few major problems or bugs during the process of designing and building this unit. For the physical construction of the unit, the only problem that arose involved the carousel. The carousel rests on a vertical, wooden rod that is connected to the motor gear. As such, when the processor steps the motor gear the rod is rotated 15˚ which, in turn, also rotates the carousel to the next pill slot. However, the team discovered a problem with the rod not being perfectly vertical. This caused some swiveling of the carousel that resulted in the auto-positioning sensor to not always be activated properly. However, this problem was easily resolved with some manual bending of the can to compensate for the imperfection on the rod's positioning. On the software side, the limited amount of memory within the PIC chip forced the team to truncate some of the original design parameters. The team had originally hoped to produce a unit that could program all 21 pill slots. However, the iteration in the code that was required to achieve this was very taxing on the memory space. Therefore, the team decided that instead of cutting some of the other design features, it would be prudent to simply allow for he programming of a few slots to demonstrate the concept. This decision allowed for the continued development of all other features to maintain robustness within the unit design.

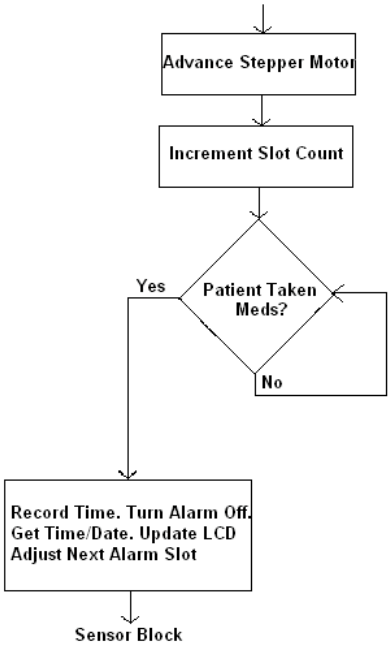
7.3 Software Logic Flowcharts

The program for the medication dispenser consisted of a loop with three main routines.

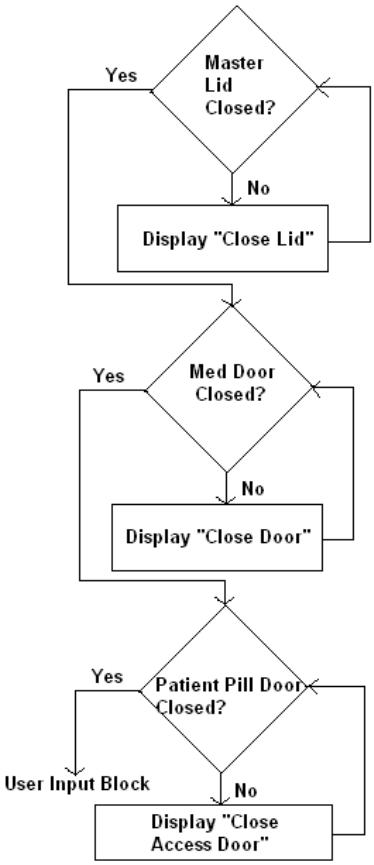


The Timer Related Block Logic is as seen below:

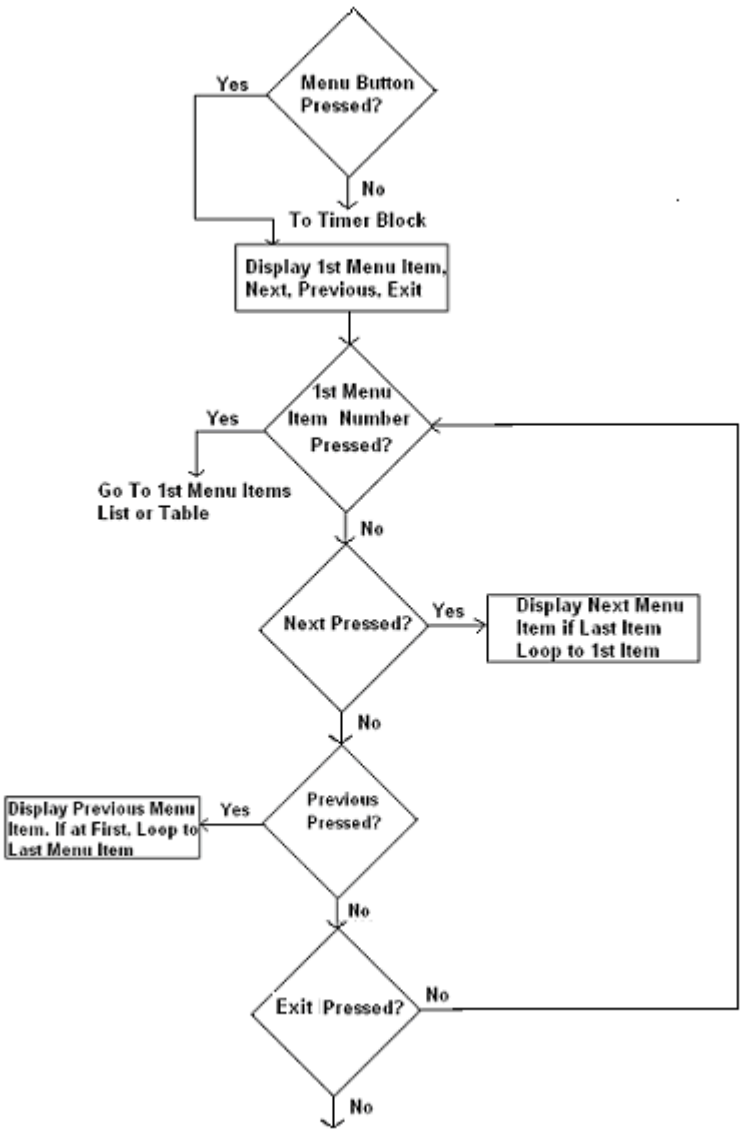




The Sensor Block Logic is as follows:



Input User Block Logic:



VIII. Appendix

8.1 **PMD Code**

```
Cntr VAR BYTE                   'This holds the index that will be used to read out of EEProm

Cntr=1
CntrIndex VAR BYTE              'This holds the index to control the number of reads at beginning of program stored at $01
                                'AlmPntr stored at $00


EEPromIndex VAR BYTE    'This holds the EEProm storage address
EEPromIndex = $02
Seconds         VAR   BYTE
Minutes         VAR   BYTE[21]      'The actual time is memory spot 0, alarm 1st slot is memory spot 1 with the remaining 20 following.
Hours  VAR BYTE[21]
Date    VAR   BYTE[21]
Month VAR   BYTE[21]
Year    VAR BYTE[21]
Prectn  VAR BYTE[21]            'Array for tracking precaution to use as messsage
AlmFlag VAR BIT[21]             'Flag for tracking if alarm has been set and if alarm has gone off
                                'This array may not work because I can't initialize the values and if power is lost
                                'any initialization routine would undo the set flags. I can initialize the values when the alarm slots
                                'are set and that may work out?

AlmOn           VAR BIT
AlmSetMin VAR BYTE[21]          'Holds the minutes that the alarm was set for
AlmSetHour VAR BYTE[21]         'Holds the hours that the alarm was set for

MinMedsTaken VAR BYTE [21]      'Memory to record minutes when patient took meds
HrMedsTaken VAR BYTE[21]        'Memory to record hours when patient took meds

TempHours     VAR BYTE    'This variable is used by the hours setting routine to strip the 12/24 and AM/PM bits off.
                                'Additionally, it will be used to hold the difference between the current time meds taken and
                                'the next alarm time.
TimeHours     VAR BYTE    'This variable is used to retain the hours once the 12/24 and the AM/PM bits are stripped off
```

```
CurrentAlm     VAR BYTE    'This will hold the current alarm for comparison
NextAlarm      VAR BYTE    'This will hold the next alarm for comparison
CurrentAlmAmPm VAR BIT'For indicating if the time is AM or PM for adjusting the
time between alarms.
NextAlmAmPm VAR BIT              'For indicating if the time is AM or PM for
adjusting the time between alarms.
CurrentAlmTens VAR BIT    'For indicating if the time has a 10 hour bit this is needed
because of the registers format
NextAlmTens VAR BIT              'For indicating if the time has a 10 hour bit this is
needed because of the registers format
DiffBtwnHrs   VAR BYTE    'This will hold the difference between the two alarm times
for adjusting the second alarm time
CurrentAlmDay         VAR BYTE    'This will hold the current alarm's day for
comparison
NextAlmDay  VAR BYTE              'This will hold the next alarm's day for comparison
NewAlmTime VAR BYTE    'This will hold the new alarm time
DiffDay                VAR BIT


ReviewSlot     VAR BYTE              'Variable for reviewing when patient took meds by
pointing to the memory spot being viewed
MinTimeBtwnMeds VAR BYTE      'This variable holds the selected amount of times
between meds


TempNum       VAR BYTE                    'This variable holds and returns the 2 digit
number from the Keys subroutine
TempNum2 VAR BYTE[2]          'For the temp storage of entered 2 digit numbers
SecDigVAR BIT
Digit VAR BYTE                            'This variable is used by the Keys subroutine
to track
                                                  'the number of digits entered TO

control the Return function
Key     VAR BYTE
TempAlmIndx VAR BYTE          'This variable is used by next slot to track slot being
set to update the TimeAlmIndx which is reset
                                                  'by the menu each time it is entered
TimeAlmIndx VAR BYTE          'This displays the slot number of the current slot
being programmed see the day, hour, etc.
AlmPntr         VAR BYTE                  'This points at the current alarm waiting to
go off

SettingAlm VAR BIT                'This variable indicates if the alarm is being set
DisplayPrctn VAR BYTE             'This bit is used to determine which precaution to
display when subroutine is called
```

```
MenuIndex VAR BYTE                  'This variable is used by the MenuChoice subroutine to select Next, Back, or Esc
MtrIndex VAR BIT                    'This is the index of the motor driver subroutine

alarm1 VAR    BIT



SettingAlm=0                'This variable is used to block steps to write to the d1307 if setting alarm 0= not setting alarm 1=set alm



Seconds=$00
Minutes[0]=$58
Hours[0]=$71
TempHours=$00
TimeHours=$00

TempNum=$00
SecDig=0
Digit=$00
Key=$00

Date[0]=$01
Month[0]=$01
Year[0]=$06
Year[1]=$00         'Temp solution to avoid alarm going off on startup
alarm1=0
AlmPntr=$01         'Starts the program with the alarm pointer pointing at the first memory slot


Pause 1000

TRISD=%11110000   'This sets D0-D3 for keypad input from microprocessor and D4-D7 for keypad output to uP
TRISB=%11000000   'This sets B7-B6 for input programming and B5-B4 and B2-B0 for motor driver functions, B3 for LCD.
TRISC=%11100011   'This sets C7-C5 for sensor input C4-C3 are for I2C communications, C2 is alarm, C1-C0 safety input.
PORTB.0=0    'This sets motor drive lines high and chip enable lines low.
PORTB.1=1
PORTB.2=1
PORTB.3=1    'LCD line
PORTB.4=1
```

```
PORTB.5=1
PORTC.2=1    'Alarm module line

'Once the code is fully written this code can be deleted

'I2CWrite PORTC.4, PORTC.3, $d0, $00, [Seconds]          'Setting the CH bit7 to 0
ensures oscillator operation.
'Pause 50

'I2CWrite PORTC.4, PORTC.3, $d0, $01, [Minutes[0]]
'Pause 50

'I2CWrite PORTC.4, PORTC.3, $d0, $02, [Hours[0]]
'Pause 50

'I2CWrite PORTC.4, PORTC.3, $d0, $04, [Date[0]]
'Pause 50

'I2CWrite PORTC.4, PORTC.3, $d0, $05, [Month[0]]
'Pause 50

'I2CWrite PORTC.4, PORTC.3, $d0, $06, [Year[0]]
'Pause 50

'End of deletion, the code can be deleted because we don't want the value in the clock
chip being reinitialized with each power up.
'Without this section when the unit is repowered the clock will need to be set the first
time but everytime after that the time should
'be retained by the D1307 chip.


PORTD.0=1              'This section initializes the values for the keypad
PORTD.1=1
PORTD.2=1
PORTD.3=1

AlmOn=0

Read $00,AlmPntr
Read $01,CntrIndex
StartLoop: IF (CntrIndex>=Cntr) AND (EEPromIndex<$FF) Then
Read EEPromIndex, Prectn[Cntr]
EEPromIndex=EEPromIndex+1
Read EEPromIndex,Minutes[Cntr]
EEPromIndex=EEPromIndex+1
Read EEPromIndex, Hours[Cntr]
```

```
EEPromIndex=EEPromIndex+1
Read EEPromIndex,Date[Cntr]
EEPromIndex=EEPromIndex+1
Read EEPromIndex,Month[Cntr]
EEPromIndex=EEPromIndex+1
Read EEPromIndex,Year[Cntr]
EEPromIndex=EEPromIndex+1
Cntr=Cntr+1
GoTo StartLoop                                  'Increments the counter
EndIF


MainLoop:   I2CRead PORTC.4, PORTC.3, $d0, $00, [Seconds]
            I2CRead PORTC.4, PORTC.3, $d0, $01, [Minutes[0]]
            I2CRead PORTC.4, PORTC.3, $d0, $02, [Hours[0]]
            I2CRead PORTC.4, PORTC.3, $d0, $04, [Date[0]]
            I2CRead PORTC.4, PORTC.3, $d0, $05, [Month[0]]
            I2CRead PORTC.4, PORTC.3, $d0, $06, [Year[0]]
            TimeHours=Hours[0]&%00011111



            'IF alarm1=0 Then           'This executes as long as the alarm is not
going off
            IF (Hours[0]>=$40)AND(Hours[0]<=$52) Then LCDOut $FE,1,"Time:
AM ",HEX2 TimeHours,":",HEX2 Minutes[0],":", HEX2 Seconds


            IF(Hours[0]>=$60)AND (Hours[0]<=$72) Then LCDOut $FE,1,"Time:
PM ",HEX2 TimeHours,":",HEX2 Minutes[0],":", HEX2 Seconds


            LCDOut $FE,$C0,HEX2 Month[0],"/",HEX2 Date[0],"/",HEX2 Year[0]
            'EndIF

            Pause 300 'To cutdown on LCD flash
        '************************ Alarm Test and Set Off
**************************************************************************
*****
            IF (Minutes[0]==Minutes[AlmPntr])
AND(Hours[0]==Hours[AlmPntr])AND(Date[0]==Date[AlmPntr])AND(Month[0]==Mo
nth[AlmPntr])AND (Year[0]==Year[AlmPntr])AND (alarm1=0)AND(AlmOn=0) Then

                IF (PORTC.0=1) AND (PORTC.1=0) Then
```

```
                    DisplayPrctn=Prectn[AlmPntr]        'This sets the precaution
display variable equal to the one chosen for this alarm
                    GoSub Precautions                   'This jumps to the
precautions subroutine and displays the chosen precaution
                    AlmSetMin [AlmPntr]=Minutes[AlmPntr]    'This records the
minutes of the current alarm time
                    AlmSetHour[AlmPntr]=Hours[AlmPntr]          'This record
the minutes of the current alarm time

                    AlmOn=1

                    GoSub MtrDvr


                    PORTC.2=0           'This port turns on the alarm module
                    alarm1=1            'Temp variable to turn off alarm for test will
use Alarm Flag later
                    EndIF
                    EndIF

          '************** Alarm turn off routine
**********************************************
          IF (PORTC.1=1)AND(AlmOn=1) Then           'This executes if the
pill output door was opened turning off the alarm
          PORTC.2=1           'Turns off alarm

          I2CRead PORTC.4, PORTC.3, $d0, $01, [MinMedsTaken[AlmPntr]]
     'Records the minutes when the patient took the meds
          I2CRead PORTC.4, PORTC.3, $d0, $02, [HrMedsTaken[AlmPntr]]
     'Records the hours the patient took the meds

          Minutes[AlmPntr]=Minutes[AlmPntr]-1  'This subtracts one minute from
the current alarm to prevent alarm from going off again
                                                                        'for
this same alarm time

          AlmOn=0                   'This resets to prevent this loop from being
entered if the alarm is not going off

          LCDOut $FE,1,"Alarm off"
          Pause 2000
          GoSub Precautions
          Pause 2000

          '**********************Alarm adjustment
section**************************************************
```

```
'                   CurrentAlmAmPm=HrMedsTaken[AlmPntr]&%00100000 'This is now. If
this equals 1 then the time was PM and I need to add 12 hours to the TimeHours
'                   CurrentAlmTens=HrMedsTaken[AlmPntr]&%00010000 'This is now. If
this equals 1 then the hours has a tens component
'                   CurrentAlm=HrMedsTaken[AlmPntr]&%00001111'This gets the current
alarm time actual one's hours
'                   CurrentAlmDay=Date[AlmPntr]


                    '********************************** This code puts the current
alarm time in 24 hour format  ************************
                     '12PM in 24 hour format
'                   IF
(CurrentAlmAmPm=%00100000)AND(CurrentAlmTens=%00010000)AND(CurrentAl
m=2)Then CurrentAlm=CurrentAlm+%00001010 'add 10
                    '10PM to 11PM in 24 hour format
'                   IF
(CurrentAlmAmPm=%00100000)AND(CurrentAlmTens=%00010000)Then
CurrentAlm=CurrentAlm+%00010110 'add 22
                    '1PM to 9PM in 24 hour format
'                   IF (CurrentAlmAmPm=%00100000)Then
CurrentAlm=CurrentAlm+%00001100 'add 12
                    '12AM in 24 hour format
'                   IF
(CurrentAlmAmPm=%00000000)AND(CurrentAlmTens=%00010000)AND(CurrentAl
m=2)Then CurrentAlm=0
                    '10AM to 11AM in 24 hour format
'                   IF
(CurrentAlmAmPm=%00000000)AND(CurrentAlmTens=%00010000)AND(CurrentAl
m=0)Then
'                   CurrentAlm=10
'                   Else
'                   CurrentAlm=11
'                   EndIF
                    '1AM to 9AM in 24 hour format require no actions



'                   NextAlmAmPm=Hours[AlmPntr+1]&%00100000 'This is next. If this
equals 1 then the time was PM and I need to add 12 hours to the TimeHours
'                   NextAlmTens=Hours[AlmPntr+1]&%00010000 'This is next. If this
equals 1 then the hours has a tens component
'                   NextAlarm=Hours[AlmPntr+1]&%00001111          'This gets the next
alarm time actual one's hours
```

```
'                   NextAlmDay=Date[AlmPntr+1]


                    '********************************** This code puts the next
alarm time in 24 hour format  ************************
                     '12PM in 24 hour format
'                   IF
(NextAlmAmPm=%00100000)AND(NextAlmTens=%00010000)AND(NextAlarm=2)T
hen NextAlarm=NextAlarm+%00001010 'add 10
                    '10PM to 11PM in 24 hour format
'                   IF (NextAlmAmPm=%00100000)AND(NextAlmTens=%00010000)Then
NextAlarm=NextAlarm+%00010110 'add 22
                    '1PM to 9PM in 24 hour format
'                   IF (NextAlmAmPm=%00100000)Then
NextAlarm=NextAlarm+%00001100 'add 12
                    '12AM in 24 hour format
'                   IF
(NextAlmAmPm=%00000000)AND(NextAlmTens=%00010000)AND(NextAlarm=2)T
hen NextAlarm=0
                    '10AM to 11AM in 24 hour format
'                   IF
(NextAlmAmPm=%00000000)AND(NextAlmTens=%00010000)AND(NextAlarm=0)T
hen
'                   NextAlarm=10
'                   Else
'                   NextAlarm=11
'                   EndIF
                    '1AM to 9AM in 24 hour format require no actions

'                   DiffDay=CurrentAlmDay-NextAlmDay                'results = 0 if the days
are the same

'                   IF (DiffDay<>0)AND(NextAlmAmPm=%00000000)Then
NextAlarm=NextAlarm+24 'The next alarm is the following day and AM => +24
                    'This makes the next equation workout

'                   DiffBtwnHrs=NextAlarm-CurrentAlm          'This gets the difference
between the current alarm and the next alarm so
                    'I can now figure out if the next alarm needs adjustment

'                   IF (MinTimeBtwnMeds>=DiffBtwnHrs) Then
DiffBtwnHrs=MinTimeBtwnMeds-DiffBtwnHrs 'DiffBtwnHrs now has the adjustment
margin

'                   IF DiffBtwnHrs<>0 Then LCDOut $FE,1, "Delay Meds by ", HEX2
DiffBtwnHrs, " Hrs"
```

```
'****************** Including alarm adjustment code to even this point cause
problems with power up display and causes the alarms
'************************ malfuction in memory this code takes the code to $1E88
which is very close to the end of the available
'****************** memory therefore I suspect this code is overwriting other code
such as the stack
'**************** without extra memory we will not be able to include the alarm
adjustment feature.


'                   IF DiffBtwnHrs=0 Then GoTo AlmOK        'No adjustment to the next
alarm is necessary

                    '******* If the days are different (DiffDay not =0) and the times are both
AM or PM. The time span is >12 hrs and ***
                    '*******                                                    no alarm
adjustment is necessary therefore no actions required                          ****
'                   IF
(DiffDay<>0)AND(CurrentAlmAmPm=%00100000)AND(NextAlmAmPm=%00100000
) Then GoTo AlmOK
'                   IF
(DiffDay<>0)AND(CurrentAlmAmPm=%00000000)AND(NextAlmAmPm=%00000000
) Then GoTo AlmOK

'                   NewAlmTime=CurrentAlm+DiffBtwnHrs
                    'Future   = Now      +DiffBtwnHrs

                    '********************* Current and Next alarm Same day Current
alarm AM  New alarm time 0-11 (AM)  ************************
'                   IF (DiffDay=0)&(CurrentAlm<=11)&(NewAlmTime<=11) Then GoSub
AlmSub

                    'This step is not necessary because can't have a zero unless zero was added
which would have caused the code to skip this
                    'IF NewAlmTime=0 Then NewAlmTime=12          'This takes the 24
hour clock and converts to 12 hour clock

                    'IF NewAlmTime>9 Then
                    'NewAlmTime=NewAlmTime-10     'This strips of the 10 therefore the
result is 0 or 1
                    'NewAlmTime=NewAlmTime|%01010000 'This is now 12hr,AM,10hr,0
or 1

                    'EndIF

                    'NewAlmTime=NewAlmTime|%01000000 'This is now 12hr,AM,1-9
```

```
                'EndIF

                '************************ Current and Next alarm Same day Current
alarm AM/PM New alarm time 12-23 (PM) **********************
'               IF (DiffDay=0)AND(NewAlmTime>=12)AND(NewAlmTime<=23) Then
        '&(CurrentAlm<=11)
'               NewAlmTime=NewAlmTime-12
'               IF NewAlmTime=0 Then NewAlmTime=12'This takes the 24 hour clock
and converts to 12 hour clock

'               IF NewAlmTime>9 Then
'               NewAlmTime=NewAlmTime-10      'This strips of the 10 therefore the
result is 0,1 or 2
'               NewAlmTime=NewAlmTime|%01110000 'This is now 12hr,PM,10hr,0 or
1

'               EndIF

'               NewAlmTime=NewAlmTime|%01100000 'This is now 12hr,PM,1-9
'               EndIF

                '************************ Current and Next alarm Same day Current
alarm PM New alarm time 12-23 (PM) ***********************
                'IF (DiffDay=0)&(NewAlmTime>=12)&(NewAlmTime<=23) Then
        '&(CurrentAlm>11)
                'NewAlmTime=NewAlmTime-12     'This puts the 24 hour clock back to
12 hour format
                'IF NewAlmTime=0 Then NewAlmTime=12          'This takes the 24
hour clock and converts to 12 hour clock

                'IF NewAlmTime>9 Then
                'NewAlmTime=NewAlmTime-10     'This strips of the 10 therefore the
result is 0,1 or 2
                'NewAlmTime=NewAlmTime|%01110000 'This is now 12hr,PM,10hr,0
or 1

                'EndIF

                'NewAlmTime=NewAlmTime|%01100000 'This is now 12hr,PM,1-9

                'EndIF
                '************************ Current and Next alarm Same day Current
alarm PM New alarm time 0-11 (AM next day) ***************
'               IF (DiffDay=0)AND(NewAlmTime>=24)AND(NewAlmTime<=35) Then
        '&(CurrentAlm>11)
'               NewAlmTime=NewAlmTime-24
```

```
'               Date[AlmPntr+1]=Date[AlmPntr+1]+1               'This should
increment the date one day
'               GoSub AlmSub
                'IF NewAlmTime=0 Then NewAlmTime=12          'This takes the 24
hour clock and converts to 12 hour clock

                'IF NewAlmTime>9 Then
                'NewAlmTime=NewAlmTime-10     'This strips of the 10 therefore the
result is 0,1 or 2
                'NewAlmTime=NewAlmTime|%01010000 'This is now 12hr,AM,10hr,0
or 1

                'EndIF

                'NewAlmTime=NewAlmTime|%01000000 'This is now 12hr,AM,1-9
'               EndIF

                '************************ Current and Next alarm Diff days Current
alarm PM New alarm time 0-11 (AM next day) ***************
                'Compiler indicated that the processor ran out of memory with this section
of code above


'AlmSub: IF NewAlmTime=0 Then NewAlmTime=12        'This takes the 24 hour clock
and converts to 12 hour clock

'               IF NewAlmTime>9 Then
'               NewAlmTime=NewAlmTime-10     'This strips of the 10 therefore the
result is 0,1 or 2
'               NewAlmTime=NewAlmTime|%01010000 'This is now 12hr,AM,10hr,0
or 1

'               EndIF

'               NewAlmTime=NewAlmTime|%01000000 'This is now 12hr,AM,1-9
'               Return

        '*****************************************************************
*******************************
'AlmOK:         pause 10

                AlmPntr=AlmPntr+1
                IF AlmPntr=22 Then AlmPntr=1                'Prevents the alarm pointer
from going out of bounds
```

```
'               IF AlmPntr<TimeAlmIndx Then AlmPntr=AlmPntr+1               'This
increments the alarm pointer to point at the next alarm in line

'               IF DiffBtwnHrs<>0 Then

'               Hours[AlmPntr] = NewAlmTime
'               LCDOut $FE,1,"Hours next med",HEX2 Hours[AlmPntr]
'               Pause 3000
'               EndIF

                alarm1=0

                EndIF
'************************************************ End of Alarm Module
**********************************************************

'               IF MinMedsTaken[AlmPntr-1]<>Minutes[0] Then alarm1=0 'This will
temporarily allow the alarm to function for the second alarm

                        'it will have to be replaced by the alarm flag.This routine has a
problem

                        'with the last memory slot because the AlmPntr is not advanced
therefore

                        'it is comparing with the memory slot from previous alarm which
will always

                        'make the if condition true
                'IF AlmOn=1 Then
                'GoSub Precautions
                'Pause 2000
                'EndIF

'Loop1:
                IF PORTC.0=0 Then
                LCDOut $FE,1,"Close Main Lid"
                Pause 2000
                        'Switch Debounce time delay
                'IF PORTC.0=0 Then GoTo Loop1
                EndIF


'Loop2:
                IF (PORTC.1=1) AND (AlmOn=0) Then
                LCDOut $FE,1,"Close Output Door"
```

```
                Pause 2000
                        'Switch Debounce time delay
'               IF PORTC.1=1 Then GoTo Loop2
                EndIF

'Loop3:         IF PORTC.6=0 Then LCDOut $FE,1,"Close Pill "
'               Pause 10
                        'Switch Debounce time delay
        '       IF PORTC.6=0 Then GoTo Loop3

'Loop4:         IF PORTC.7=0 Then LCDOut $FE,1,"At first slot"
                'Pause 10
                        'Switch Debounce time delay
                'IF PORTC.7=0 Then GoTo Loop4

                MenuIndex=0

                PORTD.3=0
                IF PORTD.4=0 Then
                Pause 100                               'Switch debounce
                PORTD.3=1
                GoTo Menu                               'This enters the menu loop
                EndIF
                PORTD.3=1               'This turns off the column because menu was not
selected


                PORTD.2=0
                IF PORTD.7=0 Then  'Pound key pushed because last alarm was set
                Pause 100                               'Switch debounce
                PORTD.2=1
PillsRdy:       GoSub MtrDvr            'This will execute until slot1 is positioned to be the
next slot over hole
                IF PORTC.7=0 Then
                LCDOut $FE,1,"All alarms set"
                Pause 2000
                GoTo   MainLoop
                Else
                GoTo PillsRdy
                EndIF
                EndIF
                PORTD.2=1               'This turns off the column because the subroutine
was not selected

                Write $00,AlmPntr
```

```
                GoTo   MainLoop



'1:             Pause 10
Menu:  LCDOut $FE,1,"Set time?  Enter"
                'LCDOut $FE,$C0,"Next, Back, or Escape"

                Pause 1000              '1 sec delay to prevent last loop keypress from
making selections in this loop
                MenuIndex=1             'Initializes the menu choice subroutine's index
number
                'GoSub ChooseEnter
                PORTD.0=0
                IF PORTD.7=0 Then               'Enter was pressed
                Pause 200
                PORTD.0=1
                TimeAlmIndx=0
                PORTD.0=1
                GoTo Minute
                EndIF
                PORTD.0=1

                GoSub MenuChoice
                'PORTD.3=0
                'IF PORTD.4=0 Then GoTo Menu
                'IF PORTD.5=0 Then GoTo MedsTaken       'Back was pressed

                'IF PORTD.6=0 Then GoTo AlSlot1 'Next was pressed

                'IF PORTD.7=0 Then GoTo MainLoop        'Escape was pressed
                'PORTD.3=1

                GoTo Menu

'2:             Pause 10
AlSlot1:LCDOut $FE,1,"Set 1st slot alarm.  Enter"
                'LCDOut $FE,$C0,"Next, Back, or Escape"

                DisplayPrctn=0
                Pause 1000
        '       GoSub ChooseEnter  'Enter was pressed
                PORTD.0=0
                IF PORTD.7=0 Then
                Pause 200
```

```
                PORTD.0=1
                CntrIndex=1
                Write $01,CntrIndex
                EEPromIndex=$02     'Resets the EEPromIndex
                AlmPntr=1      'This points to the first alarm slot in memory
                TimeAlmIndx=1        'This points to the 1st alarm memory slots
                TempAlmIndx=1        'This variable is being set so that it will be
incremented correctly each time next slot is entered.

                SettingAlm=1 'This variable tells the minute, hour, day, month, and year
not to write the values entered to the D1307 chip

MtrLoop:GoSub MtrDvr
                IF PORTC.6<>0 Then GoTo MtrLoop
                LCDOut $FE,1,"At first slot "
                Pause 2000
                IF SettingAlm=1 Then GoSub PrectnChoice

                GoTo Minute

                EndIF                          'Ends the if statement for choosing to setthe
1st alarm slot
                PORTD.0=1


                GoSub MenuChoice
                'PORTD.3=0   'These would execute if the customer doesn't want to set the
first slot alarm
                'IF PORTD.4=0 Then GoTo Menu


                'IF PORTD.5=0 Then GoTo Menu     'Back was pressed
                'IF PORTD.6=0 Then GoTo NextAlm          'Next was pressed
                'IF PORTD.7=0 Then GoTo MainLoop         'Escape was pressed
                'PORTD.3=1

                GoTo AlSlot1

'3:             Pause 10
NextAlm:LCDOut $FE,1,"Set next alarm"
                Pause 1000


                PORTD.0=0
                IF PORTD.7=0 Then  'Enter was chosen
                Pause 200
```

```
                PORTD.0=1               'Turns off first column if enter was chosen
                SettingAlm=1 'This variable tells the minute, hour, day, month, and year
not to write the values entered to the D1307 chip
                TempAlmIndx=TempAlmIndx+1       'This increments the temp alarm
index each time the next slot is selected
                LCDOut $FE,1,"Setting Alm Slot",HEX2 TempAlmIndx
                'Pause 2000
                'IF TempAlmIndx=22 Then              'Out of slots therefore this ends the
option of setting further times
                'LCDOut $FE,1,"All slots have been set"
                'Pause 2000
                'GoTo MainLoop
                'EndIF                   'End out of slots if statement

                TimeAlmIndx=TempAlmIndx
                CntrIndex=TimeAlmIndx               'This advances the counter index for
the read cycle on power up
                Write $01,CntrIndex
                GoSub MtrDvr            'This advances the carousel one slot
                GoSub PrectnChoice

                GoTo Minute
                EndIF                   'Ends the enter if then statement

                PORTD.0=1               'Turns off the first column if enter was not chosen
                GoSub MenuChoice
                'PORTD.3=0   'These would execute if the customer doesn't want to set the
first slot alarm
                'IF PORTD.4=0 Then GoTo Menu


                'IF PORTD.5=0 Then GoTo AlSlot1  'Back was pressed
                'IF PORTD.6=0 Then GoTo TimeBtwnMeds          'Next was pressed
                'IF PORTD.7=0 Then GoTo MainLoop        'Escape was pressed
                'PORTD.3=1
                GoTo NextAlm

'4:                      Pause 10
TimeBtwnMeds:LCDOut $FE,1,"Set minimum time"
                         LCDOut $Fe,$C0,"between meds"
                         Pause 1000
                         PORTD.0=0  'Turns on the first column of keys
                        IF PORTD.7=0 Then  'Enter was pressed

                        PORTD.0=1               'Turns off the enter key
                        Pause 100
```

```
                    LCDOut $FE,1,"Enter min number of"
                    LCDOut $Fe,$C0,"hours 0-12 btwn doses"
                    GoSub Keys
                    MinTimeBtwnMeds=TempNum
                    LCDOut $FE,1,"Hours chosen is",HEX2 MinTimeBtwnMeds
                    Pause 3000                          'Gives time to read the
message
                    IF (MinTimeBtwnMeds<0)|(MinTimeBtwnMeds>7)Then GoTo
TimeBtwnMeds  'Entry was out of bounds
                    GoTo MainLoop
                    EndIF
                    PORTD.0=1           'Turns off the enter key because enter wasn't
chosen
                    GoSub MenuChoice
                     'PORTD.3=0  'These would execute if the customer doesn't want
to set the first slot alarm
                    'IF PORTD.4=0 Then GoTo Menu


                    'IF PORTD.5=0 Then GoTo NextAlm  'Back was pressed
                    'IF PORTD.6=0 Then GoTo ReviewAlmTime        'Next was
pressed
                    'IF PORTD.7=0 Then GoTo MainLoop       'Escape was pressed
                    'PORTD.3=1
                    GoTo TimeBtwnMeds


'5:                  Pause 10
ReviewAlmTime: LCDOut $FE,1,"View Alarm Times"
                    'LCDOut $Fe,$C0,"Enter,Next,Back,Esc"
                    Pause 1000
                    PORTD.0=0    'Turns on the first column of keys
                    IF PORTD.7=0 Then  'Enter was pressed

                    PORTD.0=1
                    Pause 1000
                    LCDOut $FE,1,"Enter slot to review"

                    GoSub Keys
                    ReviewSlot=TempNum
                    IF ReviewSlot>21 Then GoTo ReviewAlmTime

                    TimeHours=Hours[ReviewSlot]&%00011111
```

```
                    IF (Hours[ReviewSlot]>=$40)AND(Hours[ReviewSlot]<=$52)
Then LCDOut $FE,1,HEX2 ReviewSlot,"Slot Set: AM ",HEX2 TimeHours,":",HEX2
Minutes[ReviewSlot]
                    IF(Hours[ReviewSlot]>=$60)AND (Hours[ReviewSlot]<=$72)
Then LCDOut $FE,1,HEX2 ReviewSlot,"Slot Set: PM ",HEX2 TimeHours,":",HEX2
Minutes[ReviewSlot]
                    LCDOut $FE,$C0,HEX2 Month[ReviewSlot],"/",HEX2
Date[ReviewSlot],"/",HEX2 Year[ReviewSlot]

                    Pause 3000
                    GoTo MainLoop
                    EndIF
                    PORTD.0=1    'Turns off the enter key if enter was not pressed

                    GoSub MenuChoice
                    'PORTD.3=0   'Turns on the 4th column of keys

                    'IF PORTD.5=0 Then GoTo TimeBtwnMeds          'Back was
pressed
                    'IF PORTD.6=0 Then GoTo MedsTaken                    'Next
was pressed
                    'IF PORTD.7=0 Then GoTo MainLoop
        'Escape was pressed
                    'PORTD.3=1
                    GoTo ReviewAlmTime


'6:              Pause 10
MedsTaken:LCDOut $FE,1,"View time meds taken"
                    Pause 1000

                    PORTD.0=0    'Turns on the first column of keys
                    IF PORTD.7=0 Then  'Enter was pressed

                    PORTD.0=1
                    Pause 1000
                    LCDOut $FE,1,"Enter slot to review"

                    GoSub Keys
                    ReviewSlot=TempNum
                    'IF ReviewSlot>21 Then GoTo MedsTaken


                    TimeHours=AlmSetHour[ReviewSlot]&%00011111
```

```
                    IF
(AlmSetHour[ReviewSlot]>=$40)AND(AlmSetHour[ReviewSlot]<=$52) Then LCDOut
$FE,1,HEX2 ReviewSlot,"Set: AM ",HEX2 TimeHours,":",HEX2
AlmSetMin[ReviewSlot]
                    IF(AlmSetHour[ReviewSlot]>=$60)AND
(AlmSetHour[ReviewSlot]<=$72) Then LCDOut $FE,1,HEX2 ReviewSlot,"Set: PM
",HEX2 TimeHours,":",HEX2 AlmSetMin[ReviewSlot]

                    TimeHours=HrMedsTaken[ReviewSlot]&%00011111

                    IF
(HrMedsTaken[ReviewSlot]>=$40)AND(HrMedsTaken[ReviewSlot]<=$52) Then
LCDOut $Fe,$C0,HEX2 ReviewSlot,"Taken: AM ",HEX2 TimeHours,":",HEX2
MinMedsTaken[ReviewSlot]
                    IF(HrMedsTaken[ReviewSlot]>=$60)AND
(HrMedsTaken[ReviewSlot]<=$72) Then LCDOut $Fe,$C0,HEX2 ReviewSlot,"Taken:
PM ",HEX2 TimeHours,":",HEX2 MinMedsTaken[ReviewSlot]
                    Pause 3000
                    GoTo MainLoop
                    EndIF
                    PORTD.0=1    'Turns off the enter key if enter was not pressed


          '         GoTo ReviewLoop          'This loop executes until user presses
escape to return to Main Loop

          GoSub MenuChoice
          'PORTD.3=0   'These would execute if the customer doesn't want to set the
first slot alarm
          'IF PORTD.4=0 Then GoTo Menu


          'IF PORTD.5=0 Then GoTo ReviewAlmTime      'Back was pressed
          'IF PORTD.6=0 Then GoTo Menu                    'Next was
pressed
          'IF PORTD.7=0 Then GoTo MainLoop                    'Escape was
pressed
          'PORTD.3=1
          GoTo MedsTaken


     '*************************************************************
**********
```

```
'Below this point are all the subroutines: Time and Alarm setting, Keys, Precautions, Menu Choices


Minute:         LCDOut $FE,1,"Enter minute's"
                'Pause 2000
                'LCDOut $FE,1,HEX2 TimeAlmIndx
                Pause 1000
                GoSub Keys                      'Calls Keys subroutine after subroutine program returns here
                Minutes[TimeAlmIndx]=TempNum
                IF SettingAlm=0 Then I2CWrite PORTC.4, PORTC.3, $d0, $01, [Minutes[TimeAlmIndx]]
                'The SettingAlm variable tells this part of the code if the alarm is being set
                'if the alarm is being setthen this is skipped to avoid setting the clock with the alarm time.

SubLoop1:LCDOut $FE,1,"Is",HEX2 Minutes[TimeAlmIndx],"Correct?"
                LCDOut $Fe,$C0,"Enter=Yes and 4=No"
                Pause 200

                        PORTD.0=0
                IF PORTD.7=0 Then  'Enter was pressed for yes
                        Pause 200
                        PORTD.0=1

                        'LCDOut $FE,1,Minutes[0]
                        'LCDOut $Fe,$C0,Minutes[1]

                        'Pause 1000
                        IF SettingAlm=1 Then
                        Write EEPromIndex, Minutes[TimeAlmIndx]
                        EEPromIndex=EEPromIndex+1
                        EndIF
                        GoTo Hour
                EndIF

                IF PORTD.5=0 Then
                        Pause 200
                        GoTo Minute
                EndIF

                GoTo SubLoop1

Hour:   LCDOut $FE,1,"Enter hour's"
                Pause 200
```

```
            GoSub Keys                  'Calls Keys subroutine after subroutine
program returns here
            Hours[TimeAlmIndx]=TempNum

        SubLoop2:LCDOut $FE,1,"Is",HEX2 Hours[TimeAlmIndx],"Correct?"
                    LCDOut $Fe,$C0,"Enter=Yes and 4=No"
                    Pause 200

                    PORTD.0=0
            IF PORTD.7=0 Then  'Enter was pressed for yes
                    Pause 200

                    GoTo Subloop3

            EndIF

            IF PORTD.5=0 Then
                    Pause 200
                    GoTo Hour
            EndIF

            GoTo SubLoop2

        Subloop3: LCDOut $FE,1,"AM=1 and PM=4"
            Pause 200
            IF PORTD.4=0 Then  'AM was pressed
                    Pause 200
                    TempHours=$40
                    LCDOut $FE,1,"AM was selected"
                    Pause 1000
                    GoTo Subloop4
            EndIF

            IF PORTD.5=0 Then
                    Pause 200
                    TempHours=$60
                    LCDOut $FE,1,"PM was selected"
                    Pause 1000
                    GoTo Subloop4
            EndIF
            GoTo Subloop3

        Subloop4: PORTD.0=1             'Disables the first column of numbers
        Hours[TimeAlmIndx]=TempHours|Hours[TimeAlmIndx]
        IF SettingAlm=0 Then I2CWrite PORTC.4, PORTC.3, $d0, $02,
[Hours[TimeAlmIndx]]
```

```
        'The SettingAlm variable tells this part of the code if the alarm is being set
                'if the alarm is being setthen this is skipped to avoid setting the clock with
the alarm time.
        IF SettingAlm=1 Then
                        Write EEPromIndex, Hours[TimeAlmIndx]
                        EEPromIndex=EEPromIndex+1
                        EndIF


Day:    LCDOut $FE,1,"Enter Day"
                Pause 200
                GoSub Keys                              'Calls Keys subroutine after
subroutine program returns here
                Date[TimeAlmIndx]=TempNum
                IF SettingAlm=0 Then I2CWrite PORTC.4, PORTC.3, $d0, $04,
[Date[TimeAlmIndx]]
                'The SettingAlm variable tells this part of the code if the alarm is being set
                'if the alarm is being setthen this is skipped to avoid setting the clock with
the alarm time.

        SubLoop5:LCDOut $FE,1,"Is",HEX2 Date[TimeAlmIndx],"Correct?"
                        LCDOut $Fe,$C0,"Enter=Yes and 4=No"
                        Pause 200

                        'GoSub ChooseEnter
                        PORTD.0=0
                IF PORTD.7=0 Then  'Enter was pressed for yes
                        Pause 200
                        PORTD.0=1

                IF SettingAlm=1 Then
                        Write EEPromIndex, Date[TimeAlmIndx]
                        EEPromIndex=EEPromIndex+1
                        EndIF

                        GoTo Mnth
                EndIF

                IF PORTD.5=0 Then
                        Pause 200
                        GoTo Day
                EndIF
                GoTo SubLoop5


Mnth:  LCDOut $FE,1,"Enter Month"
```

```
		Pause 200
		GoSub Keys				'Calls Keys subroutine after
subroutine program returns here
		Month[TimeAlmIndx]=TempNum
		IF SettingAlm=0 Then I2CWrite PORTC.4, PORTC.3, $d0, $05,
[Month[TimeAlmIndx]]
		'The SettingAlm variable tells this part of the code if the alarm is being set
		'if the alarm is being setthen this is skipped to avoid setting the clock with
the alarm time.

SubLoop6:LCDOut $FE,1,"Is",HEX2 Month[TimeAlmIndx],"Correct?"
		LCDOut $Fe,$C0,"Enter=Yes and 4=No"
		Pause 200

		PORTD.0=0
		IF PORTD.7=0 Then  'Enter was pressed for yes
			Pause 200
			PORTD.0=1

			IF SettingAlm=1 Then
			Write EEPromIndex, Month[TimeAlmIndx]
			EEPromIndex=EEPromIndex+1
			EndIF

			GoTo Yr
		EndIF

		IF PORTD.5=0 Then
			Pause 200
			GoTo Mnth
		EndIF
		GoTo SubLoop6


Yr: LCDOut $FE,1,"Enter Year"
		Pause 200
		GoSub Keys			'Calls Keys subroutine after subroutine
program returns here
		Year[TimeAlmIndx]=TempNum
		IF SettingAlm=0 Then I2CWrite PORTC.4, PORTC.3, $d0, $06,
[Year[TimeAlmIndx]]
		'The SettingAlm variable tells this part of the code if the alarm is being set
		'if the alarm is being setthen this is skipped to avoid setting the clock with
the alarm time.

	SubLoop7:LCDOut $FE,1,"Is",HEX2 Year[TimeAlmIndx],"Correct?"
```

```
                LCDOut $Fe,$C0,"Enter=Yes and 4=No"
                Pause 200

                PORTD.0=0
            IF PORTD.7=0 Then  'Enter was pressed for yes
                Pause 200
                PORTD.0=1

                IF SettingAlm=1 Then
                Write EEPromIndex, Year[TimeAlmIndx]
                EEPromIndex=EEPromIndex+1
                EndIF

                SettingAlm=0          'This resets the alarm set to false so the time
or alarm can be changed in the future.
                GoTo MainLoop
            EndIF

            IF PORTD.5=0 Then
                Pause 200
                GoTo Yr
            EndIF
            GoTo SubLoop7

'*****************************************************************************
************************************
```

Keys: PORTD.0=0

```
            IF PORTD.4=0 Then
                Pause 200                   'switch depressed debounce
                TempNum2[Digit]=$01
                Key=$01
            EndIF

            IF PORTD.5=0 Then
                Pause 200                   'switch depressed debounce
                TempNum2[Digit]=$04
                Key=$04                         'switch release debounce

            EndIF

            IF PORTD.6=0 Then
                Pause 200                   'switch depressed debounce
```

```
                TempNum2[Digit]=$07
                Key=$07                         'switch release debounce
        EndIF

        'IF PORTD.7=0 Then Key="*"
        IF (PORTD.4=0)OR(PORTD.5=0)OR(PORTD.6=0)Then
                KLoop1:         LCDOut $FE,1,HEX Key
                Pause 200                       'give the lcd time to
update
                IF (PORTD.4=0)OR(PORTD.5=0)OR(PORTD.6=0) Then GoTo
KLoop1
                Digit=Digit+1
                Pause 200                       'switch release
debounce
        EndIF
        PORTD.0=1



        PORTD.1=0
        IF PORTD.4=0 Then
                Pause 200               'switch depressed debounce
                TempNum2[Digit]=$02
                Key=$02
        EndIF
        IF PORTD.5=0 Then
                Pause 200               'switch depressed debounce
                TempNum2[Digit]=$05
                Key=$05
        EndIF

        IF PORTD.6=0 Then
                Pause 200               'switch depressed debounce
                TempNum2[Digit]=$08
                Key=$08
        EndIF
        IF PORTD.7=0 Then
                Pause 200               'switch depressed debounce
                TempNum2[Digit]=$00
                Key=$00
        EndIF
                IF
(PORTD.4=0)OR(PORTD.5=0)OR(PORTD.6=0)OR(PORTD.7=0) Then

                KLoop2:         LCDOut $FE,1,HEX Key
```

```
                Pause 200                                   'give the lcd time to
update
                IF
(PORTD.4=0)OR(PORTD.5=0)OR(PORTD.6=0)OR(PORTD.7=0) Then GoTo KLoop2

                Digit=Digit+1
                Pause 200       'switch release debounce
                EndIF

        PORTD.1=1


        PORTD.2=0
        IF PORTD.4=0 Then
                Pause 200                   'switch depressed debounce
                TempNum2[Digit]=$03
                Key=$03
        EndIF
        IF PORTD.5=0 Then
                Pause 200                   'switch depressed debounce
                TempNum2[Digit]=$06
                Key=$06
        EndIF
        IF PORTD.6=0 Then
                Pause 200                   'switch depressed debounce
                TempNum2[Digit]=$09
                Key=$09
        EndIF
        'IF PORTD.7=0 Then LCDOut $FE,1,"#"
        IF (PORTD.4=0)OR(PORTD.5=0)OR(PORTD.6=0) Then
        KLoop3:         LCDOut $FE,1,HEX Key
                Pause 200                                   'give the lcd time to
update
                IF (PORTD.4=0)OR(PORTD.5=0)OR(PORTD.6=0) Then GoTo
KLoop3
                Digit=Digit+1
                Pause 200
        EndIF                               'switch release debounce
        PORTD.2=1


    '   PORTD.3=0
    '   IF PORTD.4=0 Then LCDOut $FE,1,"Menu"
    '   IF PORTD.5=0 Then LCDOut $FE,1,"Back"
    '   IF PORTD.6=0 Then LCDOut $FE,1,"Next"
    '   IF PORTD.7=0 Then LCDOut $FE,1,"Escape"
```

```
'       PORTD.3=1

'       Pause 10

        IF (Digit=0)OR(Digit=1) Then
         GoTo Keys
        Else
         Digit =0
        EndIF

        TempNum= TempNum2[0]
        TempNum=TempNum<<4              'Shifts the low byte of TempNum to
the High byte of TempNum
        TempNum=TempNum|TempNum2[1]             'ORs the High byte of
temp num with the low byte of 1

        Return


MenuChoice: PORTD.3=0     'These would execute if the customer doesn't want to set the
first slot alarm

                IF PORTD.5=0 Then               'Back was pressed
                MenuIndex=MenuIndex-1
                IF MenuIndex=<0 Then MenuIndex=6
                EndIF

                IF PORTD.7=0 Then    'Escape was pressed
                PORTD.3=1
                GoTo MainLoop
                EndIF

                IF PORTD.6=0 Then               'Next was pressed
                MenuIndex=MenuIndex+1
                IF MenuIndex=>7 Then MenuIndex=1
                EndIF

                PORTD.3=1

                IF MenuIndex=1 Then GoTo Menu
                IF MenuIndex=2 Then GoTo AlSlot1
                IF MenuIndex=3 Then GoTo NextAlm
                IF MenuIndex=4 Then GoTo TimeBtwnMeds
                IF MenuIndex=5 Then GoTo ReviewAlmTime
                IF MenuIndex=6 Then GoTo MedsTaken
```

```
                    Return

'*****************************************************************
**********************************************

Precautions: Pause 100              'delay to keep lcd from flashing

                    IF DisplayPrctn=0 Then LCDOut $FE,1,"No Precautions"

                    IF DisplayPrctn=1 Then
                     LCDOut $FE,1,"Causes Drowsiness"
                     LCDOut $Fe,$C0,"Use no alcohol"
                     EndIF

                     IF DisplayPrctn=2 Then LCDOut $FE,1,"Take with water"

                     IF DisplayPrctn=3 Then LCDOut $FE,1, "Take with food"

                     IF DisplayPrctn=4 Then
                     LCDOut $FE,1,"Don't take with"
                     LCDOut $Fe,$C0,"nitrates."
                     EndIF

                     IF DisplayPrctn=5 Then LCDOut $FE,1,"May cause dizziness"

                     IF DisplayPrctn=6 Then
                     LCDOut $FE,1,"Don't use with"
                     LCDOut $Fe,$C0,"herbal products"
                     EndIF

                     IF DisplayPrctn=7 Then
                     LCDOut $FE,1,"check before using"
                     LCDOut $Fe,$C0,"with OTC Meds"
                     EndIF

                     Return


PrectnChoice:Pause 100

Subloop8:       LCDOut $FE,1,"Choose precaution"

                Pause 2000          'Time to read message
                GoSub Precautions
```

```
		Pause 2000	'Gives time to read precaution
		LCDOut $FE,1,"Is that correct"

		Pause 2000			'Gives time to read message
		PORTD.0=0		'Turns on first column of keys


		IF PORTD.7=0 Then		'Precaution was chosen and enter was
pressed
		Pause 100				'Debounce pause
		Prectn[TimeAlmIndx]=DisplayPrctn 'Enters the chosen precaution
message in the precaution array
		PORTD.0=1		'Turns off first column of keys

		IF SettingAlm=1 Then
			Write EEPromIndex, Prectn[TimeAlmIndx]
			EEPromIndex=EEPromIndex+1
			EndIF


		Return			'The precaution has been chosen and this returns the
program to the gosub call in alarm set
		'GoTo Minute
		EndIF

		PORTD.3=0		'Turns on last column of keys for precaution
selection
		'IF PORTD.4=0 Then GoTo Menu

		IF PORTD.5=0 Then DisplayPrctn=DisplayPrctn+1	'Next was pressed
			'Loops the DisplayPrctn counter around to the previous precaution

		'EndIF
		IF PORTD.6=0 Then DisplayPrctn=DisplayPrctn-1	'Back was pressed
		'Loops the DisplayPrctn counter around to the next precaution

		'EndIF
		'IF PORTD.7=0 Then GoTo MainLoop	'Escape was pressed
		PORTD.3=1			'Turns off the last column of keys
		IF (DisplayPrctn<0)Then DisplayPrctn=7
		IF (DisplayPrctn>7) Then DisplayPrctn=0
		GoTo Subloop8
```

```
MtrDvr: MtrIndex=MtrIndex+1  'This increments the index causing it to toggle each time this subroutine is called

        IF MtrIndex=1 Then
        PORTB.0=1'Chip enable clockwise
        PORTB.2=1
        PORTB.1=0  'Motor advance
        Pause 1000  'Pause to allow carousel to settle
        PORTB.1=1  'Turn off motor
        PORTB.0=0    'Turn off enable
        EndIF

        IF MtrIndex=0 Then
        PORTB.0=1'Chip enable clockwise
        PORTB.2=0
        PORTB.1=1  'Motor advance
        Pause 1000  'Pause to allow carousel to settle
        PORTB.2=1  'Turn off motor
        PORTB.0=0    'Turn off enable
        EndIF

        Return
```