Compost Master 2000 Control System Using the Motorola 68HC12 Microcontroller

Robert Todd Gordon
Gary A. Blumberg

1999 Fall Microprocessor Systems Section 2
1999 Fall Introduction to Engineering Design Section 13
December 2, 1999

TABLE OF CONTENTS

Content **Page**

Abstract 4

Introduction 5

Materials and Methods 6
1. Microcontroller *6*
2. User Interface *6*
3. Sensors *7*
4. Relay and Power Supply *10*
5. Housing *11*
6. Software *12*

Result 15

Discussion 15

Appendix A. Components 16

Appendix B. Schematics 17
1. Water Indicator *17*
2. Water Level Indicator *18*
3. Full Bin Indicator *19*
4. Full Drawer Indicator *20*
5. Temperature Indicator *21*
6. Relay Board *22*
7. LCD *23*
8. Keypad *24*

Appendix C. Software 25
1. "KEY.H" *25*
2. "CM2K.C" *29*

Bibliography 39

LIST OF TABLES

Table	Contents	Page
1.	Sensor States	10
2.	Control System Components	16

LIST OF FIGURES

Figure	Contents	Page
1.	Motorola 68HC12	6
2.	Keypad and LCD	6
3.	Moisture and Temperature Indicator	8
4.	Low Water Indicator	8
5.	Full Bin Indicator	9
6.	Full Drawer Indicator	9
7.	Relay Board	10
8.	Control System Case	11
9.	Control System Case- Interior	12
10.	Control System Case- Panels Removed	12
11.	Program Flowchart	14
12.	Water Indicator Schematic	17
13.	Low Water Indicator Schematic	18
14.	Full Bin Indicator Schematic – Receiver	19
15.	Full Bin Indicator Schematic – Transmitter	19
16.	Full Drawer Indicator Schematic	20
17.	Temperature Sensor Schematic	21
18.	Relay Board Schematic	22
19.	Liquid Crystal Display Interface	23
20.	Keypad Interface	24

ABSTRACT

The Compost Master 2000 (CM2K) offers convenient and effective compost-management to the American consumer. The Compost Master 2000 takes the work out of maintaining a rapid compost rate while reducing the some of the undesirable aspects of composting such as odor. The Compost Master 2000 takes the consumers' food and yard waste and transforms it into nutrient rich soil that can be added to their current lawn or garden.

The Motorola 68HC12 was used to implement a control system for the processing of waste in the Compost Master 2000. The microcontroller collects information from an array of sensors such as temperature and water sensors to report on the compost state. Additional sensors were used to provide the status of the receptacle itself. Infrared sensors provide a signal when either the decomposing bin or the collection drawers for completed compost is full. A water level indicator provides a signal when the water tank is low. During a processing cycle the microcontroller activates a shredder, a mixer, and a water pump (if necessary).

The user interface with the Compost Master 2000 consists of a liquid crystal display (LCD) and keypad. The liquid crystal display provides regular messages on the status of the compost and the status of the bin. The 16-button keypad is used to activate and deactivate the Compost Master 2000.

INTRODUCTION

The average American consumer produces approximately 2000 pounds of Municipal Solid Waste (MSW) per year (Bower, page 22). Yard and food wastes make up approximately 30% of the MSW stream in the United States (Compost Resource Page). Composting this particular waste stream would reduce the amount of MSW requiring disposal by almost one fourth; and provide a nutrient-rich soil amendment.

Compost added to gardens improves soil structure, texture, aeration, and water retention. When mixed with compost, clay soils are lightened, and sandy soils retain water better. Mixing compost with soil also contributes to erosion control, soil fertility, proper pH balance, and healthy root development in plants (Compost Resource Page).

Two of the largest obstacles to wide spread composting are the inconvenience and the effort of managing a compost pile effectively – both of which have technological solutions. An easy to use system that automatically maintains appropriate levels of moisture and air would accelerate the composting process and would appeal to a new consumer market. In turn, a significant step towards waste minimization would be made. Section 13 Team 1 of Introduction to Engineering Design (IED) set forth to develop the appealing, safe, informative, and effective Compost Master 2000 (CM2K). The control system, which is in part the final project for Microprocessor Systems, serves a vital role in accomplishing these goals.

MATERIALS AND METHODS

1. Microcontroller

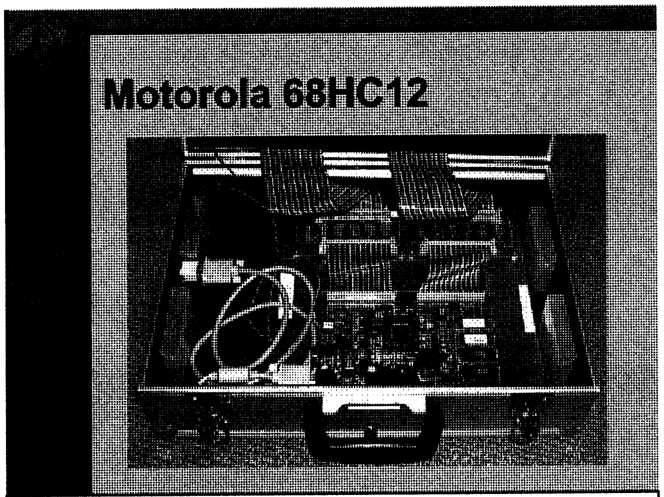


Figure 1. Motorola 68HC12

The Motorola 68HC12 was selected as the microcontroller for the control system because of previous hardware and programming experience with its predecessor, the Motorola 68HC11. The Motorola 68HC12 has four easily utilized I/O ports, which all have been used to implement the CM2K. Port G, pins 3-5, and all of port H are used to output on the LCD. The keypad uses all of port J. Pins 0-2 of port G activate the mixer, shredder, and water pump. The microcontroller receives input from sensors via port T. The 68HC12 (shown in Figure 1.) is located in the lower compartment of the electronics case and has connections to a 5 VDC power supply, a 9 pin serial connection, and two 60-pin connections (each used only in part) for the input and output with the aforementioned devices.

2. User Interface

The user interface with the CM2K consists of a keypad and LCD, which were obtained from the microprocessor lab. The LCD and keypad are mounted on the exterior of the electronics case with a protective lid. The keypad and LCD also have a sheet of transparent vinyl covering them to protect them from environmental

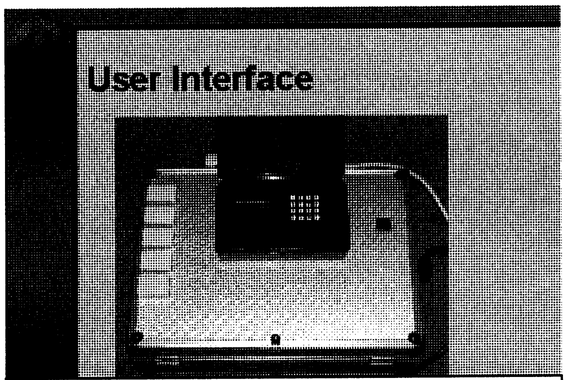


Figure 2. Keypad and LCD

elements such as water. The keypad is a Greyhill series 96 4x4 and is connected to the HC12 by the key wakeup interrupt of port J. There are sixteen keys on the keypad, the same twelve found on a telephone and the first four letters of the alphabet. The code to interface the keypad with the 68HC12 is detailed in "KEYP.H" (Appendix C–1). Eight pins (pin connections are detailed in Appendix B-8) are connected to the keypad – four are used for row selection and four are used for column selection. The LCD is a Hitachi HD44780 LCD and text is displayed up to sixteen characters in length on two separate lines. Pin connections for the LCD are detailed in Appendix B-7. The keypad allows for much wider range of input than is currently used. As such future expansions to operator input can easily be implemented for the CM2K.

3. Sensors

Radio Shack was primary source of the sensors and components, because of up-front pricing and on-line ordering. All of the major sub-components were assembly kits, which required do-it-yourself soldering (Table 2 details the component specifications). The kits, such as the infrared transmitter/receiver units, allowed focus on the overall system, instead of the very detailed level of development. Circuit boards for the kits were mounted on Plexiglas and attached to the interior of the case using Velcro (for easy removal). Three-pair telephone wire was used for internal wiring. The telephone wire bundling was convenient for pin connections on the breadboards used for circuitry. Sensor connection used two-pair and three-pair telephone wire connected to the exterior case via a single telephone jack for each sensor. The telephone wire provided easy connection to the case and weather proofing for the sensors' connection wires. The CM2K includes five sensors that report digital states. The sensors are subdivided into two groups- one group reports on the state of the compost and the other group reports on the state of receptacle.

The sensors reporting on the state of the compost are the moisture and temperature sensors. Leads on the moisture sensors are connected when the water content is sufficient to complete the circuit. The moisture sensor indicates when the compost is saturated with water. When in a connected state the microcontroller removes the water pumping stage from the mixing cycle and notifies the operator of the high water content. The temperature sensor is a thermal resistor that signals when a temperature (set via a pot on the circuit) is attained. The user is notified in case of low temperatures (decomposition slows and the water in the water tank may freeze). The circuits and the sensors are displayed in Figure 3. The schematics for the moisture and temperatures are detailed in Append ix A-1 and Appendix B-5 respectively.

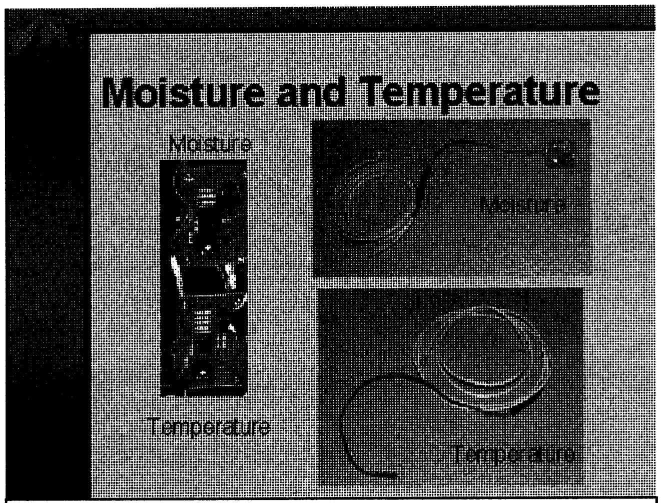


Figure 3. Moisture and Temperature Indicator

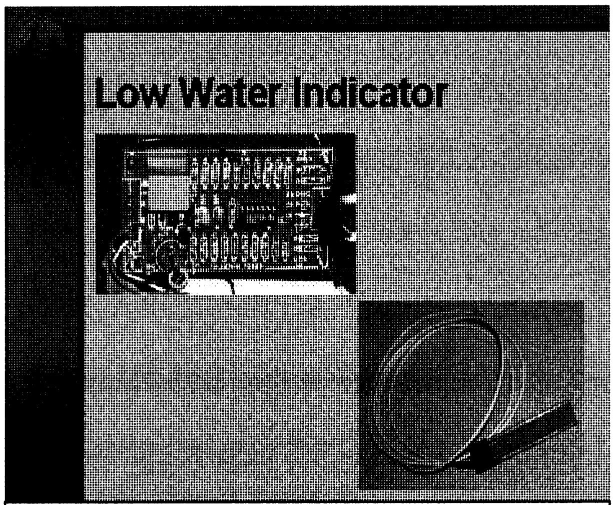


Figure 4. Low Water Indicator

The remaining three sensors report on the state of the CM2K receptacle. The low water indicator signals the microcontroller when the water level in the water tank (in the center of the CM2K) is low. The microcontroller then posts the message to refill the water tank and removes the water pumping stage from the mixing cycle. The Low Water indicator sensor and circuit are shown in Figure 4 with the schematic shown in Appendix B-2.

The full bin indicator is to be placed in a notch toward the top of the CM2K bin. The transmitter and receiver are exposed via a transparent film. Once the compost reaches a the level of the indicator the infrared transmission is blocked and a signal is sent to the microcontroller, which then disables the system since no more waste can be processed. The microcontroller then cycles a "Bin Full" message on the LCD with the any other messages that are posted. The Full Bin Indicator is shown in Figure 5 with the schematic shown in Appendix B-3.

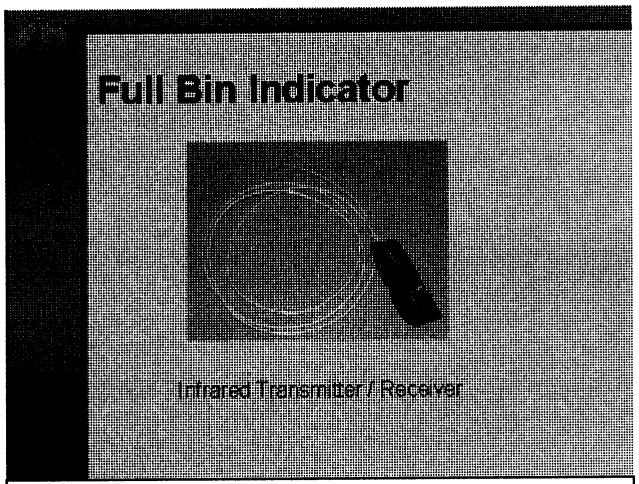


Figure 5. Full Bin Indicator

The Full Drawer Indicator is smaller than the Full Bin Indicator and has a very narrow gap between the transmitter and the receiver. The transmission is blocked by a thin black plastic sheet that is pushed into the path of the infrared LED by the pressure of the compost near the top of the drawer. When the path of the infrared is blocked a signal is sent to the microcontroller (the voltage drops to zero) and a message is cycled in the status reports to empty the drawers. The Full Bin Indicator is shown in Figure 6 with the schematic shown in Appendix B-4. The sensor states and the effects on the CM2K are summarized in Table 1.

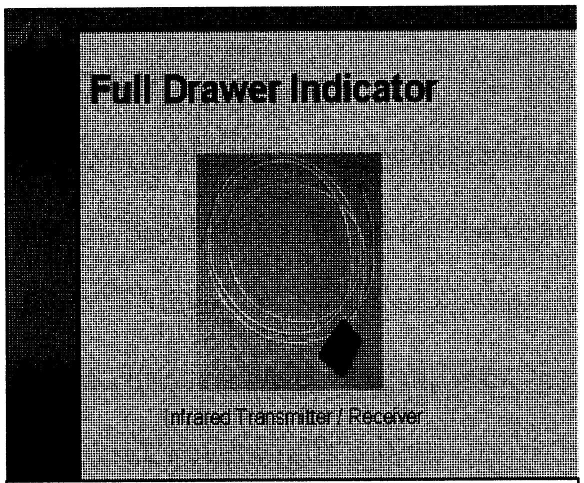


Figure 6. Full Drawer Indicator

Table 1. Sensor States

Sensor	State	Device Parameters
Bin Level	Full Not Full	System Deactivation System Activation Enabled
Drawer Level	Full Not Full	Message - Empty Drawers ------------------------------------
Moisture Content	Saturated Unsaturated	Message / Pumping Stage Disabled Pumping Stage Enabled
Temperature	Freezing Above Freezing	Message – Water May Freeze ------------------------------------
Water Level	Above Refill At or Below Refill	Pumping Stage Enabled Message / Pumping Stage Disabled

4. Relay and Power Supply

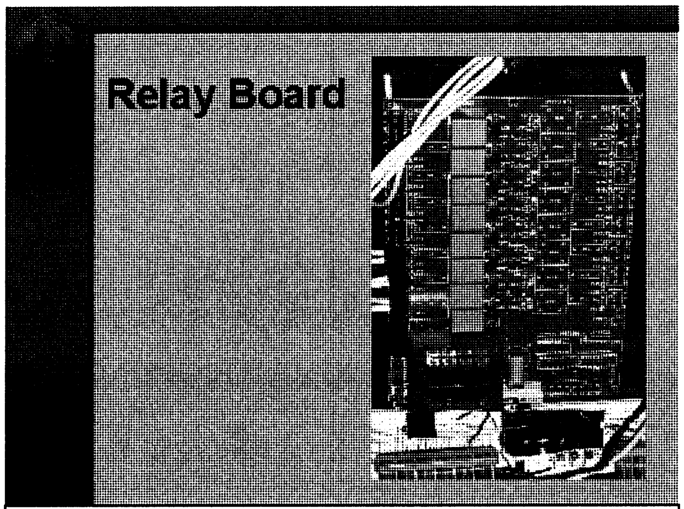


Figure 7. Relay Board

The shredder, water pump, and mixer are powered by 120 VAC. The relay board includes a transformer that converts 120 VAC to 16 VDC. The 16 VDC is used to power the low water indicator. The 16 VDC is then stepped down (using a voltage regulator) to 12 VDC to power the full bin indicator, temperature indictor, and moisture indicator. The 12 volts is then stepped down again to 5 volts to power the 68HC12 microcontroller and LCD. The 68HC12 activates the shredder, water pump, and mixer by grounding the relevant relay terminal. The power systems for the control systems electronics and the power for the external devices are optically isolated in the relays to prevent control system damage if the external devices experience current spikes.

5. Housing

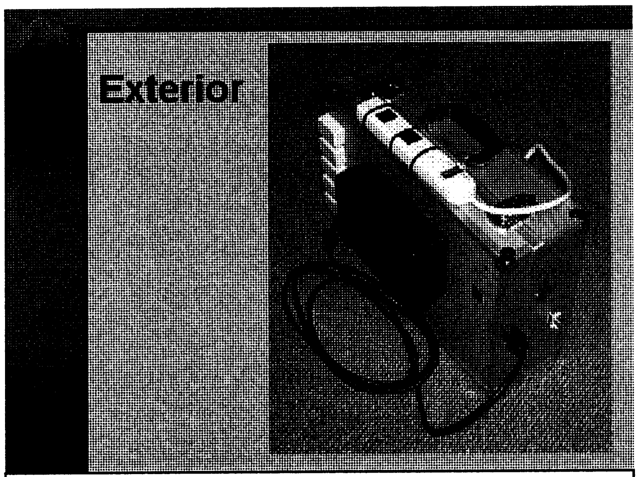


Figure 8. Control System Case

The electronics were housed in the aluminum tool case (shown in Figure 8). The sizing of the case was required for the 68HC12 Evaluation Board (from the microprocessor lab), which has an area of 1 square foot. The case also provided portability and durability for the control system. The interior consists of nonconductive foam, which was vital in prevention of short circuits related to lose wires during testing. A 9-pin serial connection port (extended from the 68HC12) was installed on the case for loading the program from a computer. Five telephone jacks were screwed into the top of the case for connection to the sensors. The black power cable plugs into 120 VAC and provides power to the control system and sensors (after a DC conversion). The black box on the lid of the case is the housing for the LCD and keypad. The red switch to the right of the keypad / LCD box is the power switch for the control systems. The switch is illuminated when the device is on. The power strip on the top of the lid is for connection to the shredder, water pump, and the mixer. The power to the outlets is wired through the relays. The devices are in an "on" state and power is supplied as appropriate to activate the devices.

As shown in Figure 9 the interior of the case has two panels separating the top of the case (where the relay and sensor circuits are located) from the bottom (where the 68HC12 board is

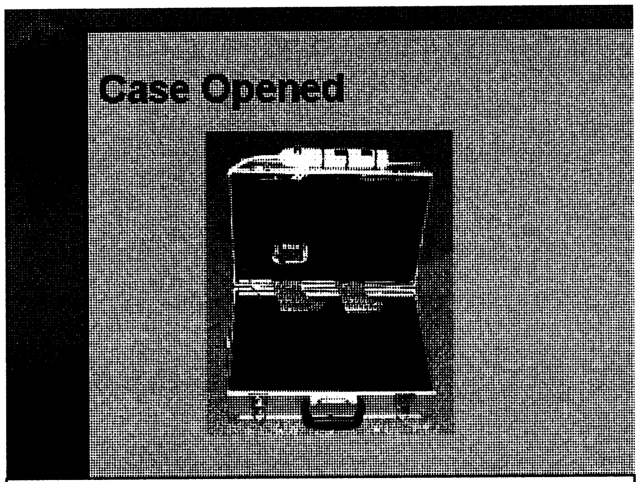


Figure 9. Control System Case - Interior

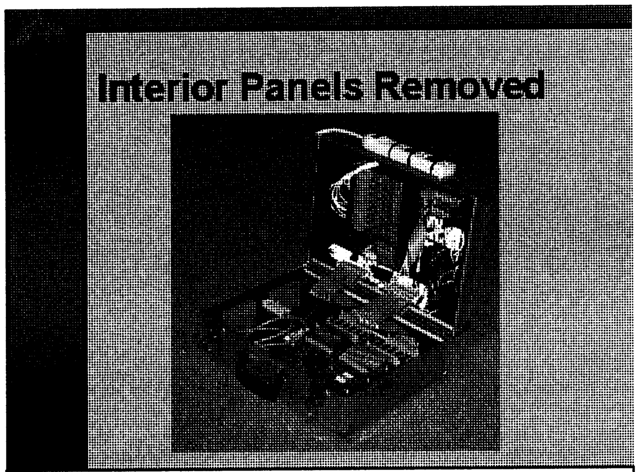


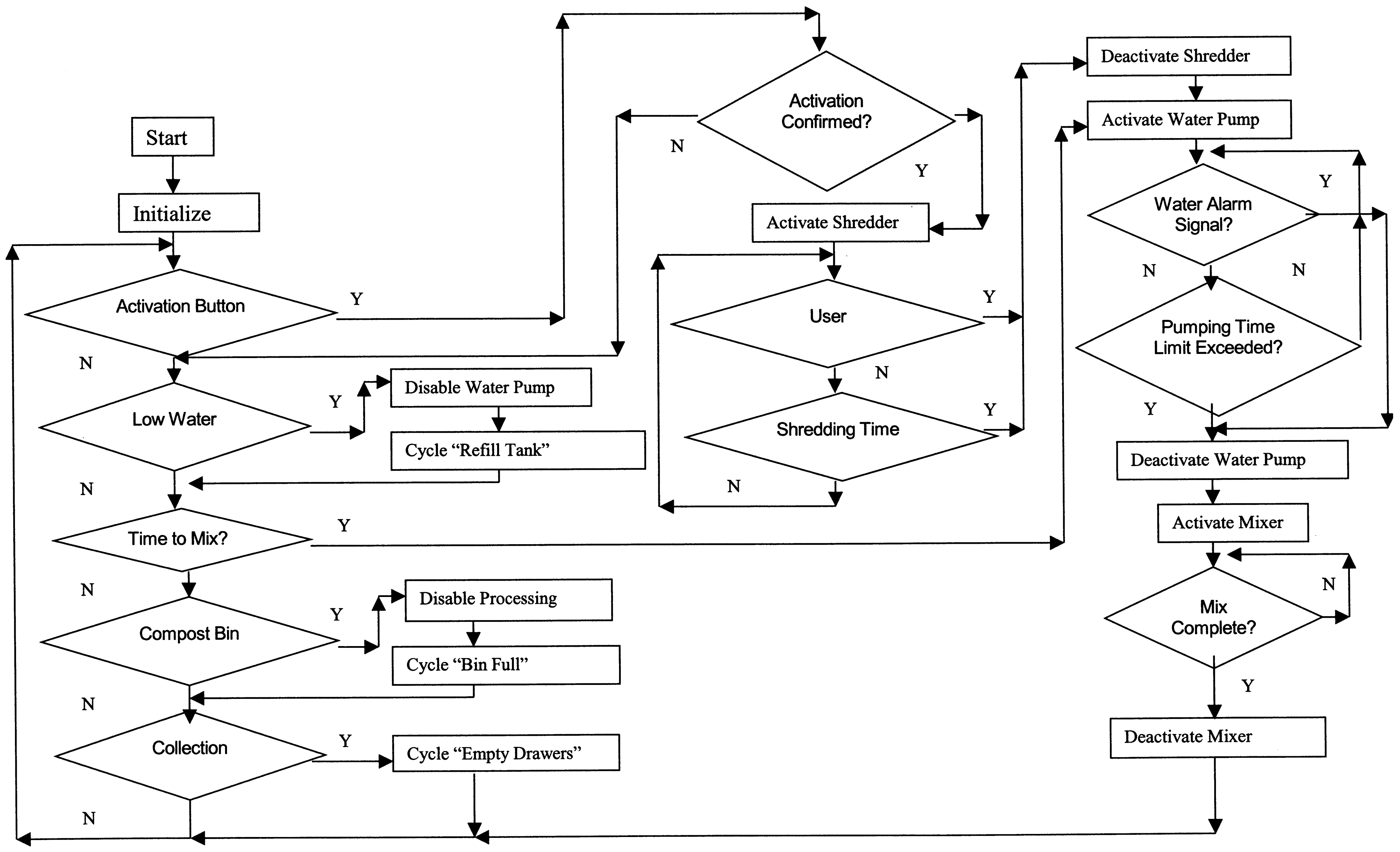
Figure 00. Control System Case - Panels Removed

located). The panels provide protection against shorting the 68HC12 with wires that may become loose from the breadboards. A twisted wire pair and two ribbon wires bridge the gap between the top and bottom compartments. The twisted wire provides the 5V power supply to the 68HC12. The ribbon wire connects to the pins of the 68HC12 EVB for input and output. The interior of the case with the panels removed is shown in Figure 10.

6. Software

The flowchart for the activation of the CM2K is detailed in Figure 11. The program consists a loop that checks the status of sensors and displays appropriate messages based on the sensor input. The LCD cycles between the messages "Compost Master 2000," "Press A to activate," and any other messages are to be displayed. The messages are timed at 2-second intervals for each message. The microcontroller waits for the operator to press "A" to active the processing cycle. As a safety precaution an operator confirmation was added to activate the system. Pressing "1" for yes activates the process, while any other key cancels the activation. For demonstration purposes the duration of each stage (shredding, water pumping, and mixing) was shortened to 6 seconds. Also, since the 12-hour mixing cycle would not be encountered during demonstrations the regular mixing cycle was excluding at this stage of development. The water pump stage is neglected if the water in the water tank is low of if the compost has high water content. In addition to the LCD display the microcontroller displays updates to the computer monitor if the computer is connected via the 9-pin serial connection. Fully documented code is included in Appendix C. "KEYP.H" is the software interface with the keypad (Appendix C-1). "CM2K.C" is the main program and its functions (Appendix C-2).

Figure 11. Program Flowchart



RESULTS

The control systems in conjunction with the other components of the Compost Master 2000 make the composting process more appealing. Maintenance of compost moisture level helps lessens the odor caused by excessive water content (which is identified by the system and countered by not adding more water). The automated features reduce the labor involved in maintaining a compost bin (regular stirring with a hoe for example is not required with the CM2K). The control system also contributes to the safety goal by curtailing accidental activation by requiring confirmation of activation.

DISCUSSION

The current control system provides a foundation upon which improved models may be developed. The decomposing environment can more ideally be maintained with more information from more sensors. Devices may be added that counter imbalances in the environment. Possible added features include analog detection of moisture and temperature. The device could also include heating elements (if they did not consume too much power) to enable cold environment composting. The water pump could also be more precisely activated for a specific amount of water to be added to the compost based on input from a water content sensor. The decomposing environment may be improved if the CM2K detected gases such as carbon dioxide and nitrogen. In addition to detecting these gases an improved model could release additives to attain a better chemical balance for the compost. The keypad could be utilized for a safety lockout code that would curtail potential accidents involving child access to the shredder. Several possibilities exist for the further development of the Compost Master 2000.

APPENDIX A. COMPONENTS

Table 2. Control System Components

Component	Selection	Cat. # **	Specifications	Qty	Unit Cost
Bin Full Indicator	Rainbow Kits Infrared Transmitter and Receiver	990-0059	Transmitter 6-15 VDC Receiver 6-12 VDC	1	$ 22.95
Keypad	Greyhill Series 96 4x4 Keypad*			1	$ 0.00
LCD	Hitachi HD44780 LCD* 10kohm potentiometer (contrast for the LCD)*		5V DC	1 1	$ 0.00 $ 0.00
Microcontroller	Motorola 68HC12 EVB*		5V DC	1	$ 0.00
Relay (Pump, Grinder, Mixer)	Universal Relay Card	990-0036		1	$ 69.95
Switch		900-7759	30A @ 12VDC		$ 2.25
Thermometer	Temperature Genie-Rainbow Kits	990-0075	Thermometer-Power: 6-15 VDC; Signal: 100Ma	1	$ 7.95
Voltage Regulator	Manufacturer LM2575T-5.0	900-8009	Input: 7V- 40V Output: 5 V	1	$ 4.89
Water Refill Indicator	Liquid Level Controller – Velleman	990-0012	12-14 VAC or 16-18VDC/100mA	1	$ 24.95
Water System Sensor	Water Alarm Kit	990-0304	Power: 7-15 VDC; Signal: 100mA	1	$ 9.99

* Provided by Microprocessor Systems Lab
** Radio Shack Electronics Catalog

APPENDIX B. SCHEMATIC 1- WATER INDICATOR

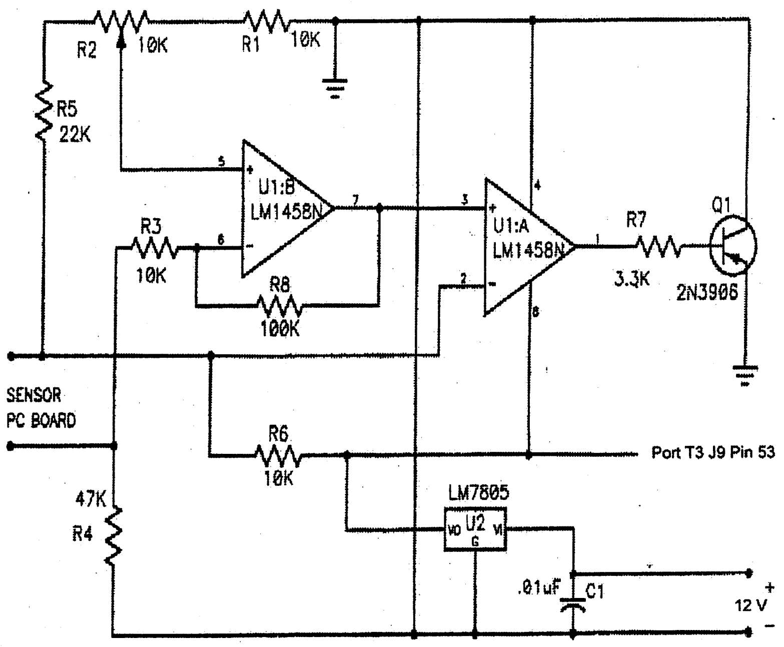


Figure 12. Water Indicator Schematic

APPENDIX B. SCHEMATIC 2- LOW WATER INDICATOR

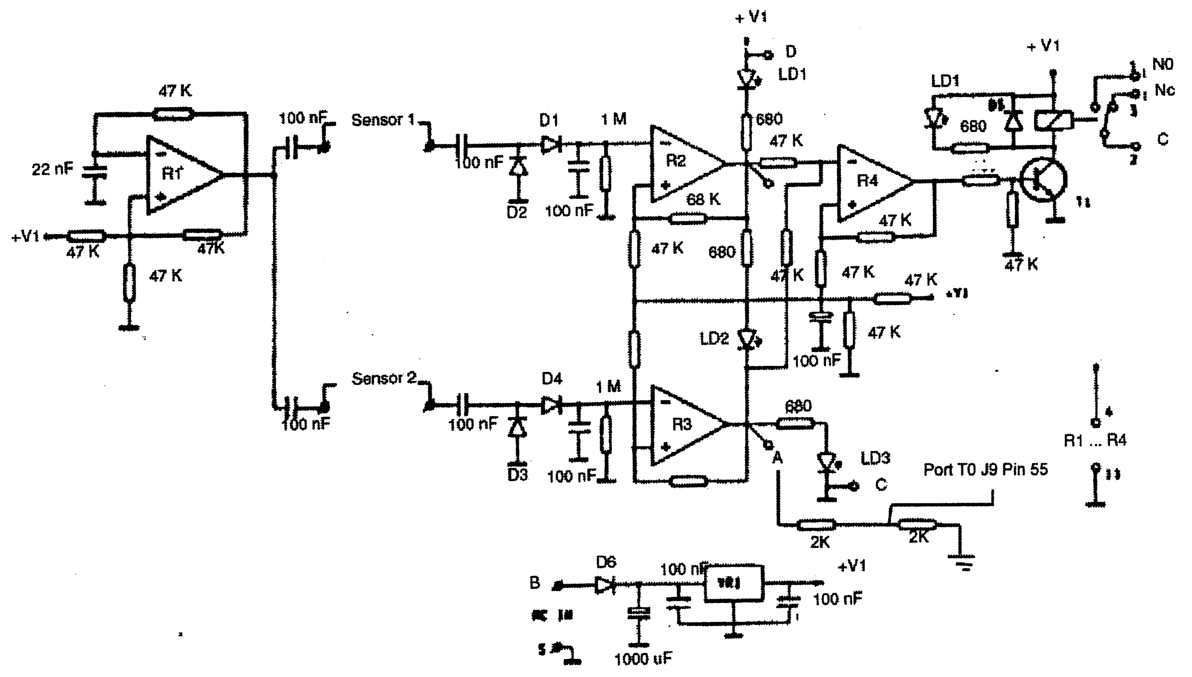


Figure 13. Low Water Indicator

APPENDIX B. SCHEMATIC 3- FULL BIN INDICATOR

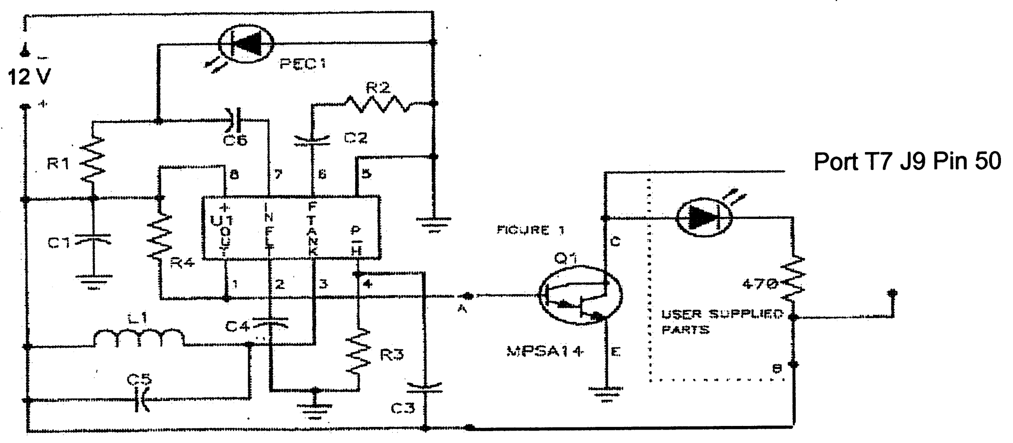


Figure 14. Full Bin Indicator - Receiver

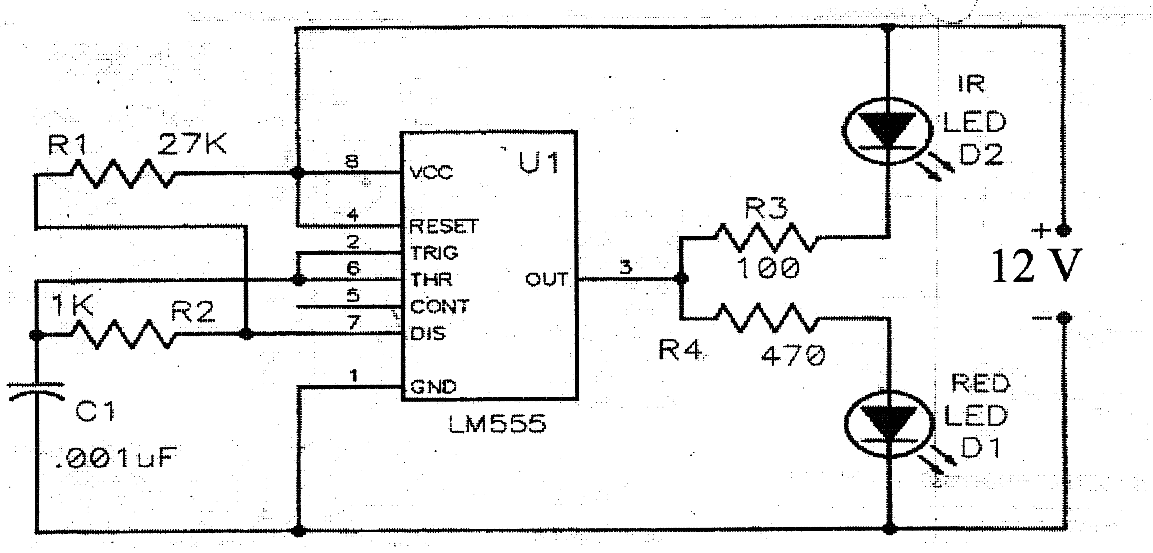
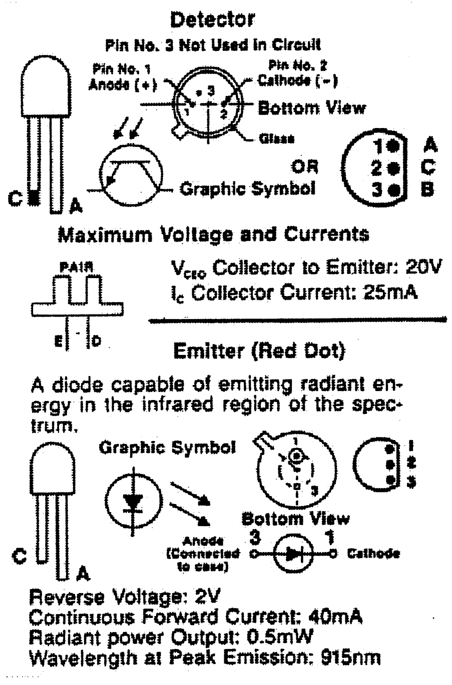


Figure 15. Full Bin Indicator - Transmitter

APPENDIX B. SCHEMATIC 4- FULL DRAWER INDICATOR



Maximum Voltage and Currents

V_{CEO} Collector to Emitter: 20V

I_C Collector Current: 25mA

Emitter (Red Dot)

A diode capable of emitting radiant energy in the infrared region of the spectrum.

Reverse Voltage: 2V
Continuous Forward Current: 40mA
Radiant power Output: 0.5mW
Wavelength at Peak Emission: 915nm

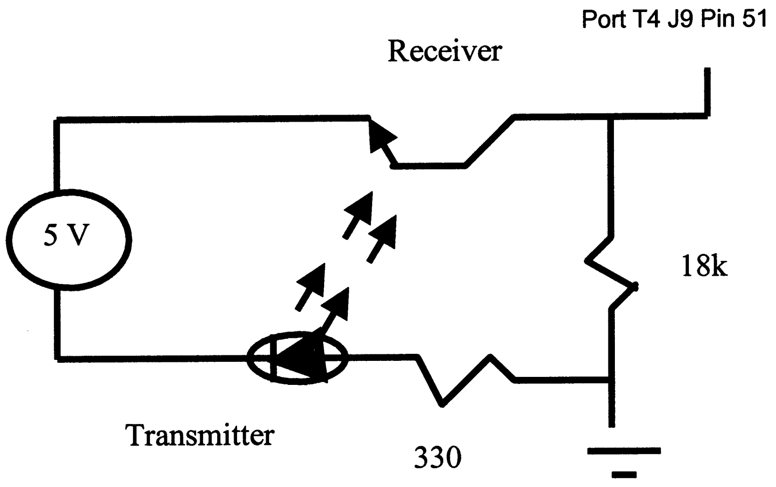


Figure 16. Full Drawer Indicator Schematic

APPENDIX B. SCHEMATIC 5 - TEMPERATURE SENSOR

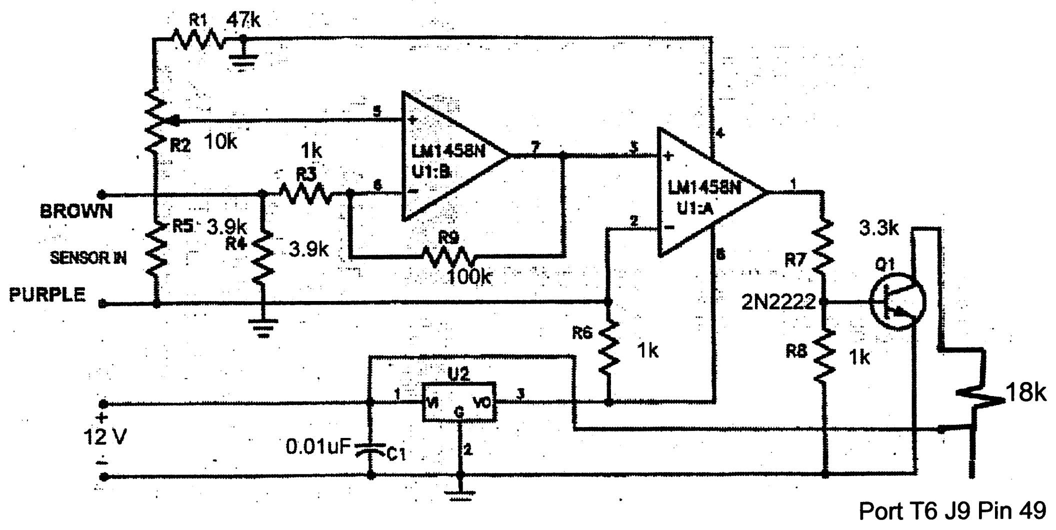


Figure 17. Temperature Sensor Schematic

APPENDIX B. SCHEMATIC 6 - RELAY

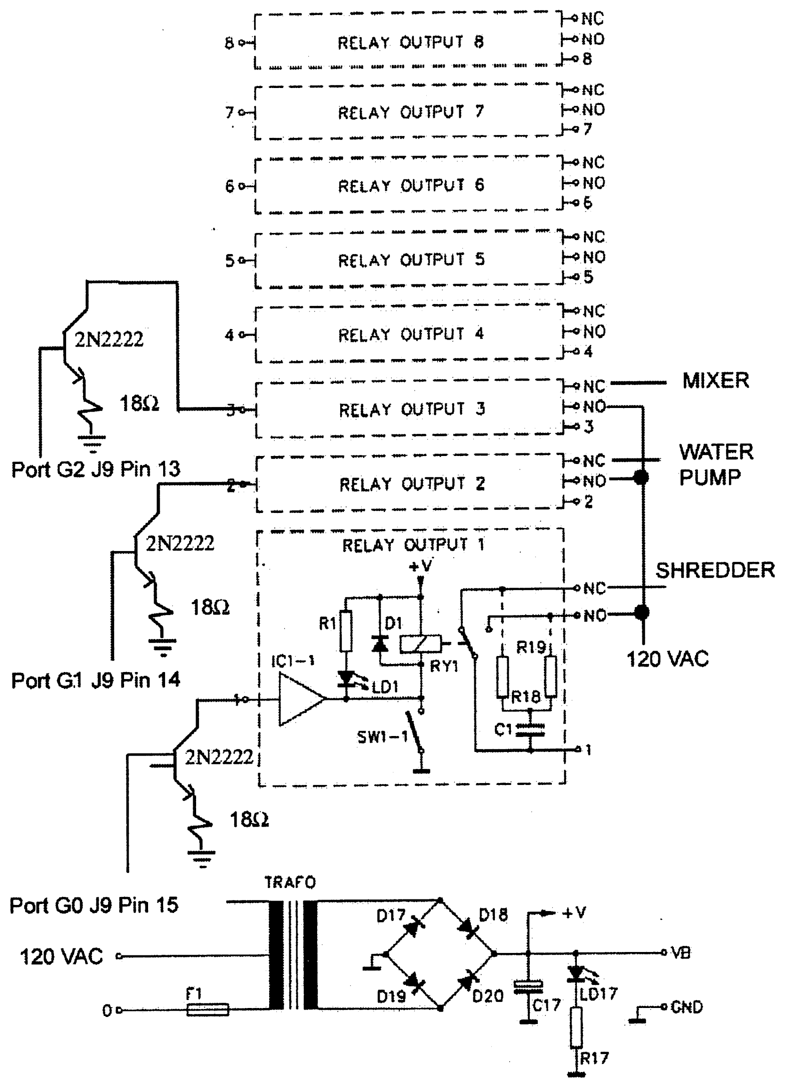
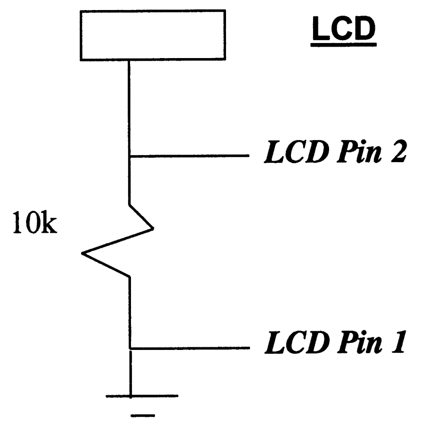


Figure 18. Relay Board Schematic

APPENDIX B. SCHEMATIC 7 - LCD

Hitachi HD44780 LCD



68HC12		**LCD**
Port G4 J8 Pin 11	X-----------------X	*LCD Pin 4*
Port G5 J8 Pin 14	X-----------------X	*LCD Pin 5*
Port G6 J8 Pin 12	X-----------------X	*LCD Pin 6*
Port H0 J9 Pin 37	X-----------------X	*LCD Pin 7*
Port H1 J9 Pin 38	X-----------------X	*LCD Pin 8*
Port H1 J9 Pin 35	X-----------------X	*LCD Pin 9*
Port H1 J9 Pin 36	X-----------------X	*LCD Pin 10*
Port H1 J9 Pin 33	X-----------------X	*LCD Pin 11*
Port H1 J9 Pin 34	X-----------------X	*LCD Pin 12*
Port H1 J9 Pin 31	X-----------------X	*LCD Pin 13*
Port H1 J9 Pin 32	X-----------------X	*LCD Pin 14*

Figure 19. Relay Board Schematic

APPENDIX B. SCHEMATIC 8 - KEYPAD

Greyhill Series 96 4x4 Keypad

68HC12		**Keypad**
Port J4 J8 Pin 3	X-----------------X	*LCD Pin1*
Port J5 J8 Pin 4	X-----------------X	*LCD Pin 2*
Port J6 J8 Pin 1	X-----------------X	*LCD Pin 3*
Port J7 J9 Pin 2	X-----------------X	*LCD Pin 4*
Port J0 J9 Pin 7	X-----------------X	*LCD Pin 5*
Port J0 J9 Pin 8	X-----------------X	*LCD Pin 6*
Port J0 J9 Pin 5	X-----------------X	*LCD Pin 7*
Port J0 J9 Pin 6	X-----------------X	*LCD Pin 8*

Figure 20. Relay Board Schematic

APPENDIX C. PROGRAM CODE 1 – "KEYP.H"

```
/* ==================================================================
   File:    keyp.h

   Gary A. Blumberg & R. Todd Gordan
   Microprocessor Systems ECSE-4790

   This file contains the key wakeup interrupt initialization routine
   and interrupt service routine used for the keypad.  The keypad is a
   Greyhill series 96 4x4 and is connected to the HC12 by the key wakeup
   interrupt of port J.  Eight pins are connected to the keypad -- four
   are used for row selection and four are used for column selection.
   ================================================================== */
#include <dbug12.h>
#include <hc812a4.h>
#include <introl.h>

// GLOBALS **********************************************************
int key;
int flag;

// FUNCTION PROTOTYPES **********************************************
void Initialize_Keypad(void);
__mod2__ void Keypad_J_ISR_Debug(void);
__mod2__ void Keypad_J_ISR(void);

// SOURCE CODE ******************************************************
void delay()
{
    long int i;
    for (i = 0; i < 1000; i++)
    {

    }
}

void Initialize_Keypad(void)
{
  // Assign vector address
  DB12->SetUserVector(PortJKey, Keypad_J_ISR_Debug);

  _H12KPOLJ = 0x00; // Sets all pins to falling edge detection

  _H12KWIFJ = 0x00; // Clear flags in case of accidental setting

  _H12PUPSJ = 0xF0; // Set pull-up / pull-down status :
                    //   7-4 pull-up   -> Columns
                    //   4-1 pull-down -> Rows

  _H12PULEJ = 0xF0; // Enable pull-up and pull-down resistors

  _H12KWIEJ = 0xF0; // Enable interrupt for column bits

  _H12DDRJ  = 0x0F; // Pull-down must be outputs
                    //   and pull-up must be inputs

  _H12PORTJ = 0x00; // Set output pins to low (should only effect 3-0)
```

APPENDIX C. PROGRAM CODE 1 – "KEYP.H"

```
  DB12->printf("Keypad initialized.\n\r");
}

// INTERRUPT SERVICE ROUTINE ****************************************
__mod2__ void Keypad_J_ISR_Debug(void)
{
  int row_bits, temp,row,column;
  flag=1;

  row_bits = _H12PORTJ;

  if (row_bits == 0x70)
  {
    row=1;
    delay();
  }
  else if (row_bits == 0xB0)
  {
    row=2;
    delay();
  }
  else if (row_bits == 0xD0)
  {
    row=3;
    delay();
  }
  else if (row_bits == 0xE0)
  {
    row=4;
    delay();
  }

    _H12PORTJ = 0x01;
    temp = _H12PORTJ;
    if ((temp & 0xF0) == 0xF0)
    {
      column=4;
      delay();
    }

      // Check for column 3
    _H12PORTJ = 0x02;
    temp = _H12PORTJ;
    if ((temp & 0xF0) == 0xF0)
    {
      column=3;
      delay();
    }

      // Check for column 2
    _H12PORTJ = 0x04;
    temp = _H12PORTJ;
    if ((temp & 0xF0) == 0xF0)
    {
        column=2;
        delay();
    }
```

APPENDIX C. PROGRAM CODE 1 – "KEYP.H"

```
    //   Check for column 1
   _H12PORTJ = 0x08;
   temp = _H12PORTJ;
   if ((temp & 0xF0) == 0xF0)
   {
       column=1;
       delay();
   }

  _H12PORTJ = 0x00;
  _H12KWIFJ = _H12KWIFJ;

  switch(row)
  {
        case 1:
                if (column == 1)
                {
                          key = 1;
                          DB12->printf("Key pressed 1\n\r");
                }
                if (column==2)
                {
                          key=4;
                          DB12->printf("Key pressed 4\n\r");
                }

                if (column==3)
                {
                          key=7;
                          DB12->printf("Key pressed 7\n\r");
                }
                if (column==4)
                {
                          key=42;
                          DB12->printf("Key pressed *\n\r");
                }
                break;
        case 2:
                if (column == 1)
                {
                          key=2;
                          DB12->printf("Key pressed 2\n\r");
                }
                if (column==2)
                {
                          key=5;
                          DB12->printf("Key pressed 5\n\r");
                }
                if (column==3)
                {
                          key=8;
                          DB12->printf("Key pressed 8\n\r");
                }
                if (column==4)
                {
                          key=0;
```

```
                        DB12->printf("Key pressed 0\n\r");
                }
                break;
        case 3:
                if (column == 1)
                {
                        key=3;
                        DB12->printf("Key pressed 3\n\r");
                }
                if (column==2)
                {
                        key=6;
                        DB12->printf("Key pressed 6\n\r");
                }
                if (column==3)
                {
                        key=9;
                        DB12->printf("Key pressed 9\n\r");
                }
                if (column==4)
                {
                        key=35;
                        DB12->printf("Key pressed #\n\r");
                }
                break;
        case 4:
                if (column == 1)
                {
                        key=65;
                        DB12->printf("Key pressed A\n\r");
                }
                if (column==2)
                {
                        key=66;
                        DB12->printf("Key pressed B\n\r");
                }
                if (column==3)
                {
                        key=67;
                        DB12->printf("Key pressed C\n\r");
                }
                if (column==4)
                {
                        key=68;
                        DB12->printf("Key pressed D\n\r");
                }
                break;
         }

}
// END OF HEADER FILE ************************************************
```

APPENDIX C. PROGRAM CODE 2 – "CM2K.C"

```
/* ==================================================================
      Gary A. Blumberg & R. Todd Gordan
      Microprocessor Systems ECSE-4790
      Course Project Program
      24 Nov 1999
==================================================================== */

/*  These three header files must be included with any code written
    for the 68HC12 using the Introl C compiler. */
#include <dbug12.h>
#include <hc812a4.h>
#include <introl.h>
#include <lcd12.h>
#include "keyp.h"

int excess_water = 0; /* compost */
int drawer_full = 0;
int bin_full = 0;
int no_water = 0; /* pump */
int seconds = 0;
int low_temp = 0;

unsigned int time_counter;
char buffer[17]; /* 16 letters max per line */

/* free running counter is 16 bits.  HC12 runs at 8x10^6 cycles/second. */
__mod2__ void TimerOverflowInt(void)
{
      time_counter++;
      if (time_counter == 122) /* very close to one second */
    {
      seconds++;
      // DB12->printf("Seconds = %d %d\n\r", seconds, seconds);
      time_counter = 0;
    }

      _H12TFLG2 = 0x80; /* clear the flag */
}

void TimeStart(void)
{
      DB12->SetUserVector(TimerOvf, TimerOverflowInt);
      time_counter = 0;
      _H12TMSK2 = 0x80; /* enable time overflow interrupt */
      _H12TSCR = 0x80;  /* enable the timer */

      DB12->printf("Timer initialized.\n\r");
}

int response(void)
{
      int temp = -1;

      WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "Press");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
```

```
        DB12->strcpy(buffer, "(1) Yes / (2) No");
        WriteBuffer(&buffer);

        flag = 0;
        key = -1;

        while(flag != 1)
        ;

        flag = 0;
        temp = key;
        key = -1;

        return temp;
}

void mix(void)
{
        int temp;

        DB12->printf("\tMixing process begun.\n\r");

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "DANGER!  Mixing");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "beginning...");
        WriteBuffer(&buffer);

        seconds = 0;
        time_counter = 0;
        while(seconds < 2)
        ;

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "Press 'D'");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "to stop.");
        WriteBuffer(&buffer);

        seconds = 0;
        time_counter = 0;
        while(seconds < 1 )
        ;

        temp = _H12PORTG;
        _H12PORTG = 0x04;

        flag = 0;
        key = -1;
        seconds = 0;
        time_counter = 0;
        while( (key != 68) && (seconds < 6) )
        ;

        flag = 0;
        key = -1;
        _H12PORTG = temp;

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "                 ");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "                 ");
        WriteBuffer(&buffer);

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "Mixing");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "completed.");
        WriteBuffer(&buffer);

        DB12->printf("\tMixing process completed.\n\r\n\r");

        seconds = 0;
        time_counter = 0;
        while(seconds < 2)
        ;
}

void pump(void)
{
        int temp;

        /* When the water supply tank is low, the pump is deactivated. */
        if( no_water == 1)
                return;

        if( excess_water == 1)
                return;

        DB12->printf("\tPumping process begun.\n\r");

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "DANGER!  Pumping");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "beginning...");
        WriteBuffer(&buffer);

        seconds = 0;
        time_counter = 0;
        while(seconds < 2)
        ;

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "Press 'D'");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "to stop.");
        WriteBuffer(&buffer);
```

```
      seconds = 0;
      time_counter = 0;
      while(seconds < 1 )
      ;

      temp =_H12PORTG;
      _H12PORTG = 0x02;

      flag = 0;
      key = -1;
      seconds = 0;
      time_counter = 0;
      while( (key != 68) && (seconds < 6) )
      ;

      flag = 0;
      key = -1;
      _H12PORTG = temp;

      WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "                    ");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
      DB12->strcpy(buffer, "                    ");
      WriteBuffer(&buffer);

      WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "Pumping");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
      DB12->strcpy(buffer, "completed.");
      WriteBuffer(&buffer);

      DB12->printf("\tPumping process completed.\n\r");

      seconds = 0;
      time_counter = 0;
      while(seconds < 2)
      ;
}

void shred(void)
{
      int temp;

      /* When the bin is full, the shredder is disabled. */
      if( bin_full == 1)
            return;

      DB12->printf("\tShredding process begun.\n\r");

      WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "DANGER! Shreding");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
      DB12->strcpy(buffer, "beginning...");
      WriteBuffer(&buffer);
```

```
        seconds = 0;
        time_counter = 0;
        while(seconds < 2)
        ;

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "Press 'D'");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "to stop.");
        WriteBuffer(&buffer);

        seconds = 0;
        time_counter = 0;
        while(seconds < 1 )
        ;

        temp = _H12PORTG;
        _H12PORTG = 0x01;

        flag = 0;
        key = -1;
        seconds = 0;
        time_counter = 0;
        while( (key != 68) && (seconds < 6) )
        ;

        flag = 0;
        key = -1;
        _H12PORTG = temp;

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "                ");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "                ");
        WriteBuffer(&buffer);

        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "Shredding");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "completed.");
        WriteBuffer(&buffer);

        DB12->printf("\tShredding process completed.\n\r");

        seconds = 0;
        time_counter = 0;
        while(seconds < 2)
        ;
}

void security(void)
{
        int input = 0;
```

```
      DB12->printf("System activated.\n\r\n\r");

    WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "Comfirm");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
      DB12->strcpy(buffer, "activation?");
      WriteBuffer(&buffer);

      seconds = 0;
      while(seconds < 2)
      ;

      input = response();

      if( input == 1)
      {
            DB12->printf("System activation confirmed.\n\r\n\r");

            shred();
            pump();
            mix();
      }
      else /*if ( input == 0) */
      {
            WriteCmdXLCD(0x01); /* clear display */
            DB12->strcpy(buffer, "Compost Master");
            WriteBuffer(&buffer);
            WriteCmdXLCD(0xC0); /* next line */
            DB12->strcpy(buffer, "NOT Activated");
            WriteBuffer(&buffer);

            seconds = 0;
            time_counter = 0;
            while(seconds < 2 )
            ;

            /* intentionally doing nothing */
      }
}

void polling(void)
{
      int temp;

      flag = 0;

      while(1)
      {
            while(flag != 1)
            {
                  seconds = 0;

                  WriteCmdXLCD(0x01); /* clear display */
                  DB12->strcpy(buffer, "Compost");
                  WriteBuffer(&buffer);
```

```
WriteCmdXLCD(0xC0); /* next line */
DB12->strcpy(buffer, "Master 2000");
WriteBuffer(&buffer);

while( (flag != 1) && (seconds < 2) )
;

DB12->printf("Status report:\n\r");

if( bin_full != 1)
{
      WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "Press 'A'");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
      DB12->strcpy(buffer, "to activate.");
      WriteBuffer(&buffer);

      DB12->printf("\tBin level - \t\tok.\n\r");

      seconds = 0;
      while( (flag != 1) && (seconds < 2) )
      ;
}
else
{
      WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "Can't process");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
      DB12->strcpy(buffer, "waste. Bin full.");
      WriteBuffer(&buffer);

      DB12->printf("\tBin level - \t\tfull.\n\r");

      seconds = 0;
      while( (flag != 1) && (seconds < 2) )
      ;
}

if( no_water == 1)
{
      WriteCmdXLCD(0x01); /* clear display */
      DB12->strcpy(buffer, "Low water tank.");
      WriteBuffer(&buffer);
      WriteCmdXLCD(0xC0); /* next line */
      DB12->strcpy(buffer, "Add water now.");
      WriteBuffer(&buffer);

      DB12->printf("\tWater tank level - \tlow.\n\r");

      seconds = 0;
      while( (flag != 1) && (seconds < 2) )
      ;
}
else
      DB12->printf("\tWater tank level - \tok.\n\r");
```

```
if(drawer_full == 1)
{
        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "Drawers are ");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "full.  Empty.");
        WriteBuffer(&buffer);

        DB12->printf("\tDrawer level - \t\tfull.\n\r");

        seconds = 0;
        while( (flag != 1) && (seconds < 2) )
          ;
}
else
        DB12->printf("\tDrawer level - \t\tok.\n\r");

if(excess_water == 1)
{
        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "High water");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "content.");
        WriteBuffer(&buffer);

        DB12->printf("\tCompost moisture level  high.\n\r");

        seconds = 0;
        while( (flag != 1) && (seconds < 2) )
          ;
}
else
        DB12->printf("\tCompost moisture level  ok.\n\r");

if(low_temp == 1)
{
        WriteCmdXLCD(0x01); /* clear display */
        DB12->strcpy(buffer, "Low Temperture");
        WriteBuffer(&buffer);
        WriteCmdXLCD(0xC0); /* next line */
        DB12->strcpy(buffer, "Water May Freeze");
        WriteBuffer(&buffer);

        DB12->printf("\tTemperaure - \t\tlow.\n\r\n\r");

        seconds = 0;
        while( (flag != 1) && (seconds < 2) )
          ;
}
else
        DB12->printf("\tTemperaure - \t\tok.\n\r\n\r");

/*
DB12->printf("Port T is ");
```

```
DB12->out2hex(_H12PORTT);
DB12->printf("\n\r");
*/

_H12PORTT = 0x00;
temp = _H12PORTT;
if( (temp & 0x01) == 0x01)
{
      no_water = 1;
      // DB12->printf("zero high\n\r");
}
else
      no_water = 0;

/* Pin one is not used in this project
_H12PORTT = 0xFF;
temp = _H12PORTT;
if( (temp & 0xFFC) == 0xFD)
      DB12->printf("\tone low\n\r");
*/

_H12PORTT = 0x00;
temp = _H12PORTT;
if( (temp & 0x04) == 0x04)
{
      // DB12->printf("\t\ttwo high\n\r");
      excess_water = 1;
}
else
      excess_water = 0;

/* Pin three is not used in this project
_H12PORTT = 0x00;
temp = _H12PORTT;
if( (temp & 0x08) == 0x08)
      DB12->printf("\t\t\tthree high\n\r");
*/

_H12PORTT = 0x00;
temp = _H12PORTT;
if( (temp & 0x10) == 0x10)
{
      // DB12->printf("\t\t\t\tfour high\n\r");
      drawer_full = 0;
}
else
      drawer_full = 1;

/* Pin five is not used in this project
_H12PORTT = 0x00;
temp = _H12PORTT;
if( (temp & 0x20) == 0x20)
{
      // DB12->printf("\t\t\t\tfive high\n\r");
}
*/
```

```
                        _H12PORTT = 0x00;
                        temp = _H12PORTT;
                        if( (temp & 0x40) == 0x40)
                        {
                                // DB12->printf("\t\t\t\t\t\t\tsix high\n\r");
                                low_temp = 0;
                        }
                        else
                                low_temp = 1;

                        _H12PORTT = 0x00;
                        temp = _H12PORTT;
                        if( (temp & 0x80) == 0x80)
                        {
                                // DB12->printf("\t\t\t\t\t\t\tseven high\n\r");
                                bin_full = 1;
                        }
                        else
                                bin_full = 0;

                }
                flag = 0;

                if(key == 65)
                        security();
        }
}

void __main()
{
        int x = 0;
    int grind=0;

    DB12->printf("Initializing system.\n\r");

        _H12DDRG = 0xFF; /* port G set output */
    /* _H12DDRG = 0x00; input direction.  If this is set there is
                no voltage change on the pins. */
        _H12DDRTT = 0x00; /* port T set for input */

    OpenXLCD(0x3F);      /* initialize screen to 5*8 double line. */
        WriteCmdXLCD(0x80); /* set address to zero */
        WriteCmdXLCD(0x0c); /* turn on display and cursur */

    Initialize_Keypad(); /* initialization */
        TimeStart();

        polling();

    WriteCmdXLCD(0x08); /* Turn off Display */
        DB12->printf("End of program.\n\r");
}
```

BIBLIOGRAPHY

1. Bower, Michael; Leon, Warren. *The Consumer's Guide to Effective Environmental Choices: Practical Advice from the Union of Concerned Scientists*. New York: Three Rivers Press, 1999.

2. The Compost Resource Page. Hp. Date [February 1, 1999]. Online. The Compost Resource Page. Available: http://www.oldgrowth.org/compost/general.html. 14 Sept. 1999.