Simulating Manufacturing Processes in Education and Training: Color Sorting Example

Richard Johnson
rjohnson@bsu.edu
Department of Industry and Technology
Ball State University

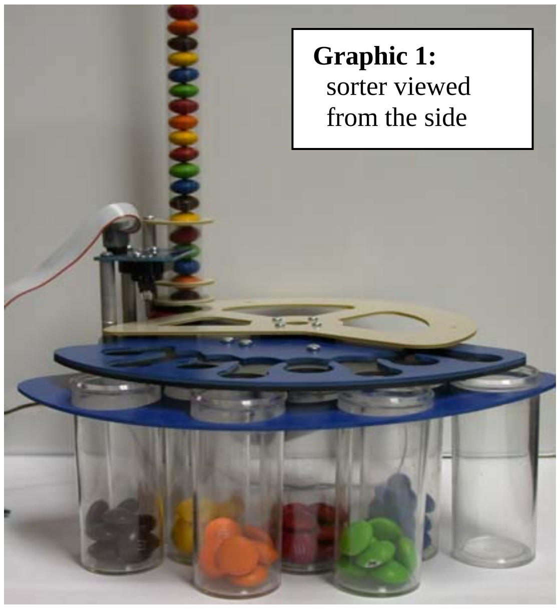
Samuel Cotton Ph.D.
scotton@bsu.edu
Department of Industry and Technology
Ball State University

Abstract

Simulation of industrial operations has become a well established and effective instructional tool used in both industrial training and formal educational programs. Duplicating large scale operations is not a cost effective strategy for training and education of trainees. Since typical operations used in manufacturing are basically the same regardless of the scale of the equipment, one can educate learners for specific tasks using smaller and less expensive equipment in a way that these skills can be easily transferred to larger operations. This article demonstrates one task, color sorting, which is typical in commercial applications that can be simulated effectively in a small scale. With decreasing budgets for education and training it is becoming critical to educate learners with tools and equipment that are affordable and do not interfere with normal company operations. Many variations are possible once the general concept of small scale industrial process simulation is mastered.

Introduction

Simulating realistic manufacturing, assembly, or transportation operations used in industry can be a valuable learning experience for students. In occupational program areas, students are often taught using applied educational methods. Accurately replicating operations, although often on a smaller scale than used in industry, is an important tool to help learners to recognize and appreciate real-world applications of new skills or concepts to be learned. To learn to operate many types of equipment, it is not always necessary to work with equipment or materials on the same scale used in industry.


Graphic 1: sorter viewed from the side

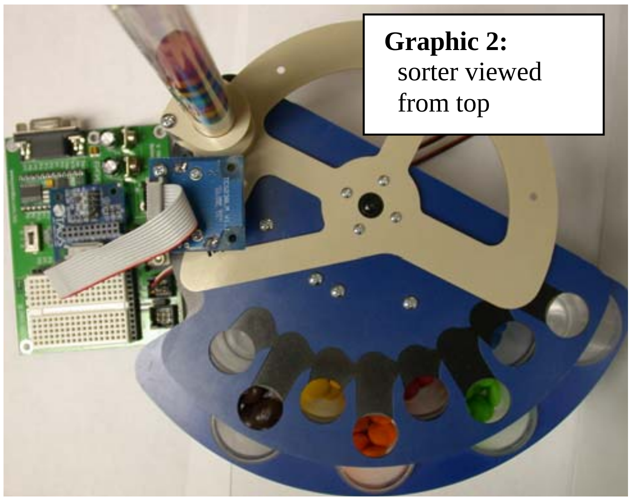
Since the skills and functions used are very similar or identical regardless of actual scale, schools or adult training centers can often effectively train future workers for the skills needed on smaller and less expensive equipment, minimizing the need to use production equipment for training. This can be attractive to industry because of reduced training expenses caused by a slow down or interruption of production work that would occur during training exercises. Schools find accurate simulations attractive because of the reduced cost of the equipment and space required for the simulations.

In both industry and education today, many production educational tasks are accomplished through computer controlled devices or operations. This reduces the need for large scale or large quantities of production equipment for either training or production. An example of an operation typical to commercial production is shared in this work that demonstrates how color sorting may be simulated in a classroom or training center. This is accomplished by sorting different colors M&M™ candies (see graphic #1). There are many other operations that could be reproduced using similar strategies (e.g. sorting by size or shape, weight, destination, or potential defect).

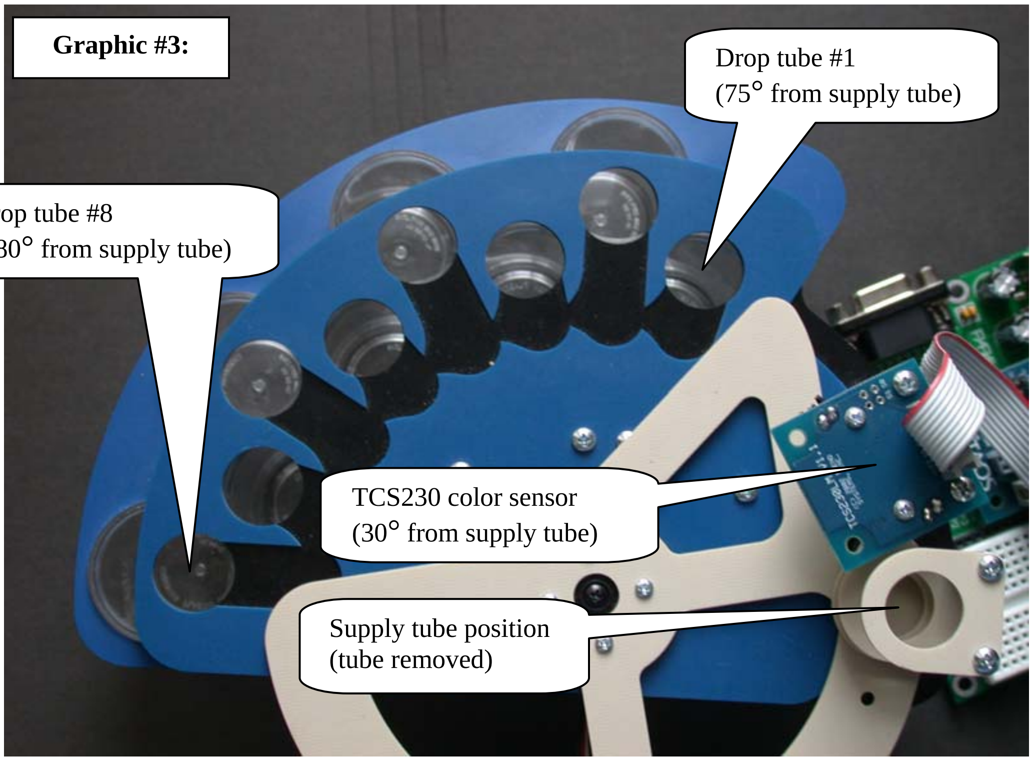
Copies of the program listings are included in appendix A for reference purposes. References to items in the program listings are included in body of the report.
For a materials list see appendix B.

Mechanical Calibration

The candy sorter used in this example will identify six different colors of candies and deposit them in an appropriate collection tube (see graphic #2). A rotating plate on the mechanism is provided to move the candies from a storage location to a color analysis location and ultimately to collection tubes that hold the candy sorted by color. One collection tube is designated for candies that the color can not be determined.

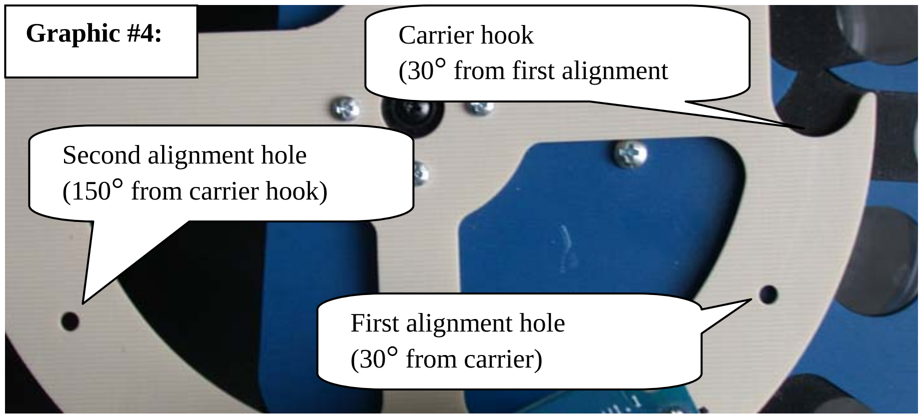

Graphic 2: sorter viewed from top

There are ten positions on the sorting mechanism (see graphic #3) that must be identified in order to operate the system. Eight positions are for the collection tubes (only seven are used for this project), one position is under the TCS230 color sensor, and one position is under the supply tube. The supply tube position is defined as the 0° or base reference point. The color sensor is 30° from the supply tube position, followed by the first drop tube at 75° from the supply tube. The other seven drop tubes are spaced at 15° increments, placing the last drop tube 180° from the supply tube.


Graphic #3:
Drop tube #1 (75° from supply tube)
Drop tube #8 (180° from supply tube)
TCS230 color sensor (30° from supply tube)
Supply tube position (tube removed)

To assist the mechanical calibration process (see graphic #4), two small alignment holes have been placed in the rotor plate, the first hole at 30° from the carrier hook and the second hole at 150° from the carrier hook. The rotor plate alignment holes can be easily detected using the TCS230 color sensor.

A simple hobby servo motor is used in this project to move rotor plate to the various positions needed to operate the sorting mechanism. The servo is controlled by sending five volts electrical signals lasting between one and two millisecond to the servo. The term “pulse width” is commonly used to describe the process of sending electrical signals


Graphic #4:
Carrier hook (30° from first alignment
Second alignment hole (150° from carrier hook)
First alignment hole (30° from carrier)

of a specific duration to a servo motor. For example, the servo used in this project will rotate to its maximum physical counter-clockwise position (see graphic #3), when an electrical signal with a pulse width of 2.0 milliseconds is sent to the servo. The servo will rotate to its maximum clockwise position when an electrical signal with a pulse width of 1.0 millisecond is sent. Pulse width signals between 1.0 millisecond and 2.0 milliseconds are used to accurately position the servo at intermediate locations. It is important to note that the pulse width values can be associated with the ten physical locations on the sorter mechanism referred to previously.

The mechanical calibration process begins by locating the second alignment hole on the rotor plate (see graphic #4); this initial rotor plate positioning is accomplished by having a human operator follow directional prompts displayed on the computer screen. Under program control the rotor plate is slowly moved while values from the color sensor are used to locate the second alignment hole. The second alignment hole location is determined when the color sensor detects a lower light level as the alignment hole moves under the sensor (see graphic #5). The lower light level results when the white LED (light emitting diode) light passes through the alignment hole on the rotor plate onto the black base plate that is located below rotor plate. The pulse width value for this alignment hole location is saved, the value is referred to as “Align_hole_2” in the program listing (see appendix A, "Program Listings"). The rotor plate is advanced and the same process is used to determine the location the first alignment hole. The servo pulse width value for the first alignment hole location is saved, the value is referred to as “Align_hole_1” in the program listing (appendix A).

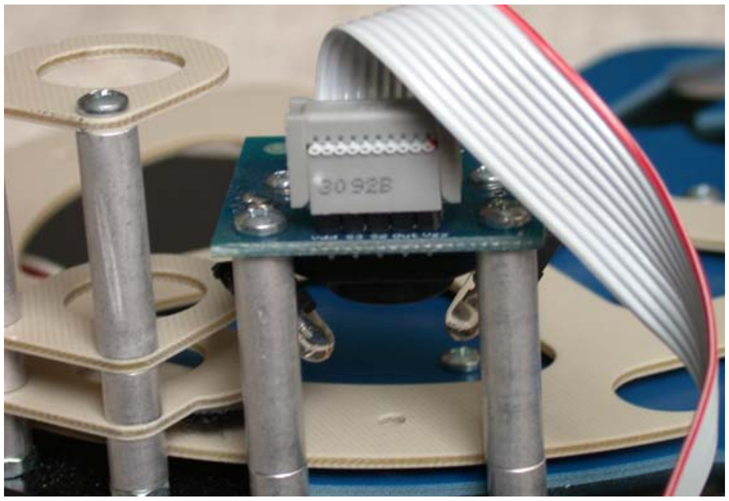
Graphic #5:

Far left: holder for the supply tube

Top center: color sensor

Bottom center: alignment hole #1

Lower right: carrier hook



After determining the location of the two alignment holes, sufficient information is available to calculate the location of the ten target positions needed to operate the sorting mechanism. For clarity all of the calculations discussed below are shown in a text box as equations. The first calculation will provide the pulse width value equivalent to 15° of rotation, multiples of the 15° rotation value can be used to compute all the other needed physical positions on the sorting mechanism. The two alignment holes are known to be 120° apart, therefore the difference between the two alignment hole pulse width values is equivalent to 120°. Dividing the difference between the two alignment hole pulse width values by 8 will give a pulse width value that is equivalent to 15°, this value is referred to as “Pos_delta” in the program listing (appendix A).

$$\text{Pos\_delta} = |(\text{Align\_hole\_1} - \text{Align\_hole\_2})| / 8$$

Calculation #1

Three pulse width values are calculated that will be used to position the rotor plate carrier hook at one of three locations: the supply tube position, the color sensor position, and the first drop tube position. The relative positions of the supply tube to the first drop tube and to the color sensor position are known to be minus 30° and plus 45° respectively. Also, the carrier hook on the rotor plate is known to be plus 30° from the first alignment hole. Calculation #2 computes the pulse width value needed to position the carrier hook under the color sensor by subtracting the pulse width equivalent of 30° from the first alignment hole pulse width value. Calculation #3 computes the pulse width value needed to position the carrier hook under the supply tube by subtracting the pulse width equivalent of 60° from the first alignment hole pulse width value. The last calculation (#4) computes the pulse width value of the first drop tube position by adding the pulse width equivalent of 15° to the first alignment hole pulse width value.

```
Color Sensor position➔      Sensor_Pos = Align_hole_1 – (Pos_delta * 2)
Supply tube position➔       Home_pos = Align_hole_1 – (Pos_delta * 4)
First drop tube position➔   Hole_1 = Align_hole_1 + Pos_delta
```

Calculations #2, #3, and #4

The pulse width values for the color sensor position, supply tube position, and first drop tube position in addition to the "Pos_delta" value are written to the Electrically Erasable Programmable Read Only Memory (EEPROM) for later use by the operation program (appendix A). EEPROM is a type of memory that can be changed, but retains its information when the power is turned off. The other seven drop tube positions are calculated in the operation program to save EEPROM space. The first time a sorting system microcontroller completes the calibration process a calibration flag value of "99" is written to the EEPROM. Table #1 shows a partial EEPROM memory map.

Table 1: Partial EEPROM Memory Map

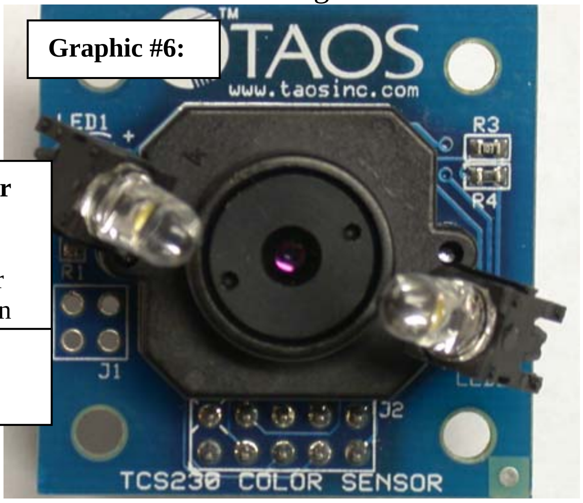
Location	**Usage**
0	Pos_delta, pulse width equivalent to 15°
2:3	First drop tube pulse width position
4:5	Color sensor pulse width position
6:7	Supply tube pulse width position
8	Calibration flag value (99 or 199)

The program code for the mechanical calibration begins at the "Arm_Calibration:" label in the calibration program listing (appendix A).

The TCS230 Color Sensor

The TCS230 sensor module (see graphic #6) from Texas Advanced Optoelectronic Solutions (TAOS) has an array of photodetectors most of which have one of three color (red, green, or blue) filters. The photodetectors are evenly distributed across the array to avoid location bias among the colors. The sensor outputs a square-wave whose frequency is proportional to the intensity of the selected color of light. The color filter selection is accomplished by pulling the S2 and S3 lines on the sensor high or low as shown in table #2. A program subroutine was developed ("Check_color" in the program listings) to activate the sensor, turn on the two white light emitting diodes, select the appropriate filter and obtain red, blue, and green color intensity values. More information about TCS230 sensor can be obtained from the Texas Advanced Optoelectronic Solutions web site [www.taosinc.com].

Graphic #6:



S2	**S3**	**Color**
0	0	Red
0	1	Blue
1	0	Clear
1	1	Green

Table 2: Color Filter Selection

Calibrating the Color Sensor System

This implementation of the M&M™ sorter system identifies six different colors of candies. The red, green, and blue (RGB) color intensity values were obtained for two candies of each color and these values averaged. The highest and lowest acceptable RGB intensity values for each color were computed and then written to the EEPROM for later use by the operation program.

The RGB intensity values were obtained experimentally for twenty candies of each color (brown, yellow, orange, red, green, and blue). The RGB intensity values were analyzed for each color to determine an acceptable range of RGB values that would accurately identify each color of candy. The guideline used in determining the acceptable RGB ranges assume it is better to classify a candies as unknown rather than sort it incorrectly. If the average color intensity value was greater than 100 then an acceptable range was determined to be plus or minus 10% of the average value for each color. However, if the average color intensity value was less than 100 then, the red color range was determined to be plus or minus 20% of the average value, the green color and blue color ranges were determined to be plus or minus 30% of the average value. These finding are summarized in table #3.

Color/Intensity	<= 100	> 100
Red	+/- 20%	+/- 10%
Green	+/- 30%	+/- 10%
Blue	+/- 30%	+/- 10%

Table 3: Color Range Values

At the completion of the color sensor calibration process a calibration flag value of "199" is written to the EEPROM. The operation program checks for this value in the EEPROM to confirm that the sorting system has completed the required calibration steps. The program code for the color sensor calibration begins at the "Color_Calibration:" label in the calibration program listing (appendix A).

Operating the M&M™ Sorter

The first task performed by the operation program is to confirm that the calibration routines have been successfully completed by verifying that the calibration flag value stored the EEPROM has been set to '199'. Next, a window is displayed on the computer instructing the operator to load the sorter supply tube with candy and then press '1' to begin the sorting process.

The sorting process begins by moving the rotor plate with the candy of unknown color in the carrier hook from the supply tube position to the color sensor position. After positioning the candy under the TCS230 color sensor, the sensor is activated and the red, green, and blue color intensity values for the unknown candy are obtained. Imbedded in a controlled, "FOR/NEXT" loop are statements used to read through RGB color intensity values of known candy colors previously saved in the EEPROM and compare those values to the unknown candy RGB color intensity values. A six condition compound "IF" statement is used to compare the unknown candy RGB intensity values to the acceptable high and low RGB intensity values computed in the calibration program. If a known candy color is found the program variable "mmColor" is assigned a value from zero to five otherwise the "mmColor" variable is assigned a value of six indicating the candy color can not be identified.

The last major task that the operation program performs is to move a candy from the sensor position to the appropriate drop tube position based on its color. The first drop tube is designated to hold candies of undetermined color, drop tubes two through seven are designated to hold one of the six known candy colors that are found. Recall that the calibration program determined the pulse width position value of the first drop tube (Align_hole_1) and the pulse width value equivalent to 15° (Pos_delta) and saved those values to the EEPROM. The drop tubes are known to be 15° apart, therefore the position of drop tubes two through seven can be computed by adding multiples of the "Pos_delta" value to the first drop tube position value. For example the position value of the third drop tube would be equal to the first drop tube position value plus two times the "Pos_delta" value.

Occasionally the supply tube becomes jammed by an oversized or irregularly shaped candy creating a processing exception condition. The system detects this condition when the color sensor is activated and the resulting RGB color intensity values are all very low (less than 10 in most cases). If a candy is not found under the sensor, the operation program moves the carrier hook to the supply tube position one more time in an attempt to "bump" a candy loose from the supply tube. If after two attempts at obtaining a candy, a candy is not moved under the color sensor it is assumed that the supply tube is jammed or the supply tube is empty. In either case a message is displayed on the computer screen requesting intervention by the operator.

Summary

Using the color sorting simulation will help learners understand the processes needed to prepare equipment to analyze and execute an operation based on color identification. By understanding this process/procedure a learner can easily transfer much of this skill into new situations that use color as a factor in an operation. Many other types of sensors require similar procedures for calibration, so the process used to make these preparations are transferable to other situations.

This type of system may be enhanced by increasing the number of alternative storage locations, using multiple sorting units, sending sorted items through additional processes instead of storage containers, etc. Color or other types of sorters are often only stations that are part of a much larger processing sequence. For simulations on smaller budgets, equipment used may be modified to replicate each step of a process instead of sequencing multiple stations in a single operation. The disadvantage of this approach is the lag time required to modify a process, but the advantages can be a dramatic reduction in expense or the ability to concentrate on individual steps in a process one at a time.

With experience and additional training, learners can begin to develop more and more sophisticated systems and can begin to combine a number of different tasks together into a larger coordinated process. By using smaller simulations, errors constitute a smaller financial impact and a broader variety of tasks can be experimented with in a shorter timeframe than might be required for large scale operations. Small simulations also require fewer personnel to operate and maintain equipment.

An enhanced version of the sorter has been developed that incorporates a small LCD (liquid crystal display) in place of the computer monitor and pushbuttons mounted on the unit replacing the computer keyboard. These enhancements provide greater portability by reducing dependence on local computer-based control. Future options under development include the integration of a board level web server, radio frequency (RF) communication devices, and streaming video. These enhancements will allow the sorting device to be monitored and to some extent remotely controlled through the World Wide Web. This will allow effective instruction for learners who are not able to be visit the classroom and operate the equipment directly. Methods and strategies for web-based control of manufacturing equipment is explored in the article “Interfacing with Manufacturing System in both Education and Industry Using Microcontrollers through the World Wide Web” (Cotton & Johnson) in this issue. [1].

Acknowledgement

The authors would like to thank Mr. Christopher A. Hileman for his technical contributions and dedication to the development of this project.

References

[1] Cotton, S., Johnson, R. (2006). Interfacing with Manufacturing System in both Education and Industry Using Microcontrollers through the World Wide Web, the Technology Interface.

Appendix A – Program Listing

This appendix contains three (3) microcontroller program listings used with the color sorting device. The calibration program is needed to calibrate the mechanical and color sensor subsystems. The operation program performs the primary sorting functions. The last program (ReadEEPROM) is a utility program used to display the value written to the EEPROM by the calibration program.

Calibration program listing
Operation program listing
ReadEEPROM Program listing

Calibration program listing for sorter

```
'{$STAMP BS2}
'{$PBASIC 2.5}
'*********************************************
'     Program = Operation Program            '
'     Purpose:  Use this program to calibrate'
'      the M&M Sorter                        '
'     Authors:  Rick Johnson & Chris Hileman '
'********************************************'

'**************************
'Program Declarations
'**************************
Servo_pin   CON   12
EN          CON   1
A0          CON   2
S0          CON   3
S1          CON   4
S2          CON   5
S3          CON   6
nLED        CON   7
OUT         CON   8

pRED        CON   12
pGREEN      CON   8
pBLUE       CON   12

RED         VAR   Word
```

```
GREEN       VAR   Byte
BLUE        VAR   Byte
Last_RED    VAR   Word
Last_GREEN  VAR   Byte
Last_BLUE   VAR   Byte
'***********************
'Calulation Declarations
'***********************
Align_Hole_2  VAR Word                  'Calibration Hole2 VAR
Align_Hole_1  VAR Word                  'Calibration Hole1 VAR
Target_Pos    VAR Word                  'Location Calculation var
ColorCal      VAR Word                  'Color Calculation VAR
Cur_pos       VAR Word                  'Current Position VAR
Next_pos      VAR Word                  'Next Position VAR
ColorCal2     VAR Word                  'Color Calculation VAR

Pos_Delta     VAR Byte                  'Delta of Position Increments
Cur_Color     VAR Byte                  'Current Color
Home_Pos      VAR Byte                  'Home Position
Sensor_Pos    VAR Align_Hole_2          'Reuse of Variable for Sensor Position
Hole_1        VAR Align_Hole_1          'Reuse of Variable for Drop Hole #1

Answer        VAR Nib                   'Menu Choice (DEBUGIN)
Cal           VAR Bit                   'Calibration flag on / off

BaseR         CON 10                    'EEPROM ADDRESS
FinishR       CON 5 * 8 + BaseR         'EEPROM ADDRESS
'************************
'End of Declarations
'************************



Cal = 0
READ 8, Cur_Color
IF Cur_Color = 99 OR Cur_Color = 199 THEN
  Cal = 1
ENDIF

Start:                                  'Start loop of Calibration User interface menus
```

```
Answer = 0
DEBUG CLS, "Welcome to Sorter Calibration", CR, CR

DEBUG "What would you like to do? ", CR
DEBUG "[1] Calibrate Control Arm", CR
IF Cal = 1 THEN
  DEBUG "[2] Calibrate Color Sensor", CR
ENDIF
DEBUGIN DEC1 Answer
DEBUG CR, DEC1 Answer
BRANCH Answer, [Start, Arm_Calibration, Color_Calibration, Start]
GOTO Start

Arm_Calibration:                        'Start of Arm Calibration routine
  Cur_pos = 800
  Answer  = 1
Arm_Cal_Loop:
DEBUG CLS, "Arm Alignment Routine", CR
DEBUG CR, "Press '1' until arm is fully rotated then press '2'", CR
DO WHILE (Answer = 1)
  Next_pos = Cur_pos + 100
  FOR Cur_pos = Cur_pos TO Next_pos STEP 5
    PULSOUT Servo_pin, Cur_pos                          '*****************************************'
    PAUSE 10                                            'Press 1 until the arm is rotated, this  '
  NEXT                                                  'to calibrate the arm starting with the  '
  DEBUGIN DEC1 Answer                                   'location of calibration hole #2 and     '
LOOP                                                    'then moving to hole #1                  '
                                                        '*****************************************'
Next_pos = Cur_pos - 200
GOSUB Check_Color
Last_RED   = RED
Last_GREEN = GREEN
Last_BLUE  = BLUE
FOR Cur_pos = Cur_pos TO Next_pos STEP 2
  PULSOUT Servo_pin, Cur_pos
  PAUSE 10
  GOSUB Check_Color
  IF (RED <= Last_RED + 1) AND (GREEN <= Last_GREEN + 1) AND (BLUE <= Last_BLUE + 1) THEN
    Last_RED   = RED
    Last_GREEN = GREEN
    Last_BLUE  = BLUE
  ELSE                                                    '**************************************************'
    Align_Hole_2 = Cur_pos                                'In this loop, it records the previous RGB values '
    EXIT                                                  'and then compares those versus the present values'
  ENDIF                                                   'If there is any drop in Number, (A Hole Located) '
NEXT                                                      'That location is then recorded for future use in '
FOR Cur_pos = cur_pos TO 500 STEP 5                       'other calculations that are to be done           '
  PULSOUT Servo_pin, Cur_pos                              '**************************************************'
  PAUSE 10
NEXT
Next_pos = Cur_pos - 200
GOSUB Check_Color
Last_RED   = RED
Last_GREEN = GREEN
Last_BLUE  = BLUE
FOR Cur_pos = Cur_pos TO 200 STEP 2
  PULSOUT Servo_pin, Cur_pos
  PAUSE 30
  GOSUB Check_Color
  IF (RED <= Last_RED + 2) AND (GREEN <= Last_GREEN + 2) AND (BLUE <= Last_BLUE + 2) THEN
    Last_RED   = RED
    Last_GREEN = GREEN
    Last_BLUE  = BLUE
  ELSE
    Align_Hole_1 = Cur_pos
    EXIT
  ENDIF
NEXT

Compute_Target_Locations:                                          'Loop Calculates Primary Locations
Pos_Delta = Align_hole_2 - Align_hole_1 * 10 / 8 + 5 / 10
WRITE 0, Pos_delta                                                 ' Write to EEPROM Position Delta
Target_Pos = Align_Hole_1 - (Pos_delta * 2)
WRITE 4, Word Target_pos                                           ' Write to EEPROM Sensor Position
Target_Pos = Align_Hole_1 - (Pos_delta * 4) + 22
WRITE 6, Word Target_pos                                           ' Write to EEPROM Tube/Home Position
Target_Pos = Align_Hole_1 + Pos_delta
WRITE 2, Word Target_pos                                           ' Write to EEPROM Drop Hole #1 Position
IF cal = 0 THEN
  WRITE 8, 99
```

```
ENDIF
Cal = 1
DEBUGIN DEC1 answer
GOTO Start


Check_Color:                            ' Read the color of subject with sensor
  LOW A0                                ' Init Sensor
  HIGH  S0
  HIGH  S1
  LOW nLED                              'LED's On
  PAUSE 200
  HIGH EN
  LOW S2                                ' Test for color
  LOW S3
  COUNT out, pRED, RED
  HIGH S3
  COUNT out, pBLUE, BLUE
  HIGH S2
  COUNT OUT, pGREEN, GREEN
  LOW EN                                ' Sensor off
  LOW nLED                              'LED's Off
RETURN
                                                        '***************************************'
Color_Calibration:                                      'Test to see if Arm has been Calibrated'
IF Cal = 0 THEN                                         'and then Calibrate color Sensor follow'
  DEBUG CR, "Please Run Arm Calibration First", CR      'On screen instructions                '
ENDIF                                                   '***************************************'

READ 0, Pos_delta
READ 2, Word Hole_1
READ 4, Word Sensor_Pos
READ 6, Home_Pos

DEBUG CR, "Please Insert 2 M&M's of each color in this order into top of tube ",CR
DEBUG "Brown, Yellow, Orange, Red, Green, Blue ",CR
DEBUG "Press 1 to continue ",CR
DEBUGIN DEC1 Answer
FOR Cur_Color = BaseR TO FinishR STEP 8
  FOR Answer = 1 TO 2 STEP 1
```

```
GOSUB Sensor_Position
GOSUB Check_Color
IF Answer = 1 THEN
  Last_Red   = Red
  Last_Blue  = Blue
  Last_Green = Green                             '*****************************************'
ELSE                                             'Uses colors that the sensor detects and  '
  IF ((Last_Red + Red) / 2) > 100 THEN           'Calculates ranges of the colors that we  '
    ColorCal   = (Last_Red + Red) * 9 / 20       'will use to compare the sensor's readings'
    ColorCal2  = (Last_Red + Red) * 11 / 20      'later.  It then writes those ranges to   '
  ELSE                                           'the proper EEPROM address                '
    ColorCal   = (Last_Red + Red) * 8 / 20       '*****************************************'
    ColorCal2  = (Last_Red + Red) * 12 / 20
  ENDIF
  WRITE Cur_Color, Word ColorCal
  WRITE (Cur_Color + 2), Word ColorCal2

  IF ((Last_Green + Green) / 2) > 100 THEN
    ColorCal   = (Last_Green + Green) * 9 / 20
    ColorCal2  = (Last_Green + Green) * 11 / 20
  ELSE
    ColorCal   = (Last_Green + Green) * 7 / 20
    ColorCal2  = (Last_Green + Green) * 13 / 20
  ENDIF
  WRITE (Cur_Color + 4), ColorCal
  WRITE (Cur_Color + 5), ColorCal2

  IF ((Last_Blue + Blue) / 2) > 100 THEN
    ColorCal   = (Last_Blue + Blue) * 9 / 20
    ColorCal2  = (Last_Blue + Blue) * 11 / 20
  ELSE
    ColorCal   = (Last_Blue + Blue) * 7 / 20
    ColorCal2  = (Last_Blue + Blue) * 13 / 20
  ENDIF
  WRITE (Cur_Color + 6), ColorCal
  WRITE (Cur_Color + 7), ColorCal2

  PAUSE 20
ENDIF
GOSUB Brown_Pos
```

```
  NEXT
NEXT
WRITE 8, 199
Cal = 1
GOSUB Sensor_Position
GOTO Start

Sensor_Position:                        'Loop to position Servo From HOME -> Sensor
  FOR Cur_pos = Home_Pos TO Sensor_Pos STEP 2
    PULSOUT Servo_pin, Cur_pos
    PAUSE 20
  NEXT
RETURN

Brown_Pos:                              'Default Dump for all M&M's (Hole #1)
  FOR Cur_pos = Sensor_pos TO Hole_1 STEP 4
    PULSOUT Servo_Pin, Cur_Pos
    PAUSE 20
  NEXT
     FOR Cur_pos = Hole_1 TO Sensor_Pos STEP 25   'Drop Hole to Sensor Fast return
      PULSOUT Servo_Pin, Cur_pos
      PAUSE 20
     NEXT
  GOSUB Check_Color
  FOR Cur_pos = Sensor_pos TO Home_Pos STEP 4    'To allow servo to return from Sensor -> Home slowly
    PULSOUT Servo_Pin, Cur_pos
    PAUSE 20
  NEXT
RETURN
```

Operation program listing for sorter

```
'{$STAMP BS2}
'{$PBASIC 2.5}

'*********************************************
'     Program = Operation_Program            '
'     Purpose:  Use this program to sort     '
'        M&M's using the Parallax M&M Sorter '
'     Authors:  Rick Johnson & Chris Hileman '
'*********************************************'

'**************************
'Program Declarations
'**************************
Sensor_Pos      VAR   Word           'Sensor_Pos = Position of the RGB Sensor
Drop_1          VAR   Word           'Pos_1 = Position of the #1 Tube
Pos_Delta       VAR   Byte           'Delta of the Position of the tube
Home_Pos        VAR   Byte           'Home_pos = Home position of Servo

RedHigh         VAR   Word           'High Red VAR
RedLow          VAR   Word           'Low Red VAR
Cur_x           VAR   Word           'Cur_X = current position of Servo
Word_Var        VAR   Word           'Variable used for different calculations in program
GreenHigh       VAR   Byte           'High Green VAR
GreenLow        VAR   Byte           'Low Green VAR
BlueHigh        VAR   Byte           'High Blue VAR
BlueLow         VAR   Byte           'Low Blue VAR

mmColor         VAR   Nib            'Variable set to signal M&M Color
Counter         VAR   Nib            'Counter Variable
A               VAR   Bit            'Variable used for multiple things

Servo_Pos       CON   12
mmColorUnknown  CON   6              'To tell Branch statement unknown Color
Move_MM_Step    CON   8              'Step Value for all Branch cases

'***************************
'EEPROM ADDRESSES
'***************************
BaseR           CON   10
FinishR         CON   5 * 8 + BaseR

'***************************
'Color Sensor Declarations
'***************************
EN              CON   1
A0              CON   2
```

```
S0              CON   3
S1              CON   4
S2              CON   5
S3              CON   6
nLED            CON   7
OUT             CON   8

pRED            CON   12
pGREEN          CON   8
pBLUE           CON   12

RED             VAR   Word
GREEN           VAR   Word
BLUE            VAR   Word
'***************************
'End of Declarations
'***************************

READ 8, Home_Pos                         'Checking for Calibration of Sorter
IF Home_Pos <> 199 THEN
  GOTO Check_Cal
ENDIF

READ 0, Pos_Delta                         'EEPROM Read for Servo positions
READ 2, Word Drop_1                       'EEPROM Read for Servo positions
READ 4, Word Sensor_Pos                   'EEPROM Read for Servo positions
READ 6, Home_Pos                          'EEPROM Read for Servo positions


MainStart:                                'Main Menu for Start of Program
  DEBUG REP "*" \ 40
  DEBUG CR
  DEBUG CR, "Welcome to the M&M Sorter, ",CR
  DEBUG "Please load tube full of M&M's and press 1",CR
  DEBUG CR
  DEBUG REP "*" \40 , CR
  DEBUGIN DEC1 A

InnerMain:                                 '*****************************************'
  GOSUB Sensor_Position                    'This is the Main control loop for this   '
  DEBUG CR, "Now Checking Color...",CR    'program.  It uses multiple sub-routines  '
  GOSUB Check_color                       'to accomplish this                       '
  DEBUG " Now Placing M&M"                '*****************************************'
  GOSUB Color_Choice
  GOSUB Drop_mm
  IF Counter > 1 THEN                     'Will check supply tube twice to see if its empty
    Counter = 0
    DEBUG CR, "Operator Intervention Required", CR
    DEBUG "Press 1 and Enter to restart", CR
    DEBUGIN DEC A
    GOTO MainStart                        'Done sorting
  ENDIF
  GOTO InnerMain

Check_color:                              '*****************************************'
   HIGH EN                                'Loop will use color sensor to check the  '
   LOW A0                                 'color and then store the values it sees  '
   HIGH  S0                               '*****************************************'
   HIGH  S1
   LOW nLED
  PAUSE 200
  LOW S2
  LOW S3
  COUNT OUT, pRED, RED
  HIGH S3
  COUNT OUT, pBLUE, BLUE
  HIGH S2
  COUNT OUT, pGREEN, GREEN
  LOW EN
  LOW nLED
  RETURN

Color_Choice:                                         '*************************************'
IF (RED < 10) AND (GREEN < 10) AND (BLUE < 10) THEN   'Loop gets information and then decides'
  mmColor = 15                                        'Whether it is a known color or unknown'
  RETURN                                              'Then uses lookdown command to set a   '
ENDIF                                                 'Value to a Variable for future use    '
mmColor = mmColorUnknown                              '*************************************'
FOR Word_Var = BaseR TO FinishR STEP 8
 READ Word_Var, Word RedLow
 READ (Word_Var + 2), Word RedHigh
 READ (Word_Var + 4), GreenLow
 READ (Word_Var + 5), GreenHigh
 READ (Word_Var + 6), BlueLow
 READ (Word_Var + 7), BlueHigh
 IF (RED >= RedLow) AND (RED <= RedHigh) AND (GREEN >= GreenLow) AND (GREEN <= GreenHigh) AND (BLUE >=
BlueLow) AND (BLUE <= BlueHigh) THEN
   LOOKDOWN Word_Var, [BaseR, BaseR + 8, BaseR + 16, BaseR + 24, BaseR + 32, BaseR + 40], mmColor
   DEBUG CR ,"Found Color "
   EXIT
 ENDIF
NEXT
RETURN
                                                 '*********************************************'
Drop_mm:                                         'Loop using Branch statement for tube location'
  IF mmColor = 15 THEN                           'of the M&M and then sends a signal to the    '
      Counter = Counter + 1                      'proper subroutine to sort the M&M            '
      GOTO Drop_mm_Exit                          '*********************************************'
  ENDIF
  DEBUG "Now Sorting M&M",CR
  Cur_x = Sensor_Pos
  BRANCH mmColor, [Brown_Pos, Yellow_Pos, Orange_Pos, Red_Pos, Green_Pos, Blue_Pos, Unknown_Pos]
  DEBUG "CRITICAL ERROR!", CR
  DEBUG "Attention Required!", CR
  GOTO Drop_mm_Exit

Unknown_Pos:                                       'Tube location for Unknown Color
  FOR Cur_x = Cur_x TO Drop_1 STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
  GOTO Moveit

Brown_Pos:                                         'Tube location for Brown M&M
  Word_Var = Drop_1 + Pos_Delta
  FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
  GOTO Moveit
```

```
Yellow_Pos:                                         'Tube location for Yellow M&M
  Word_Var = Drop_1 + (2 * Pos_Delta)
  FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
  GOTO Moveit

Orange_Pos:                                        'Tube location for Orange M&M
  Word_Var = Drop_1 + (3 * Pos_Delta)
  FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
  GOTO Moveit

Red_Pos:                                           'Tube location for Red M&M
  Word_Var = Drop_1 + (4 * Pos_Delta)
  FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
  GOTO Moveit

Green_Pos:                                         'Tube location for Green M&M
  Word_Var = Drop_1 + (5 * Pos_Delta)
  FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
  GOTO Moveit

Blue_Pos:                                          'Tube location for Blue M&M
  Word_Var = Drop_1 + (6 * Pos_Delta)
  FOR Cur_x = Cur_x TO Word_Var STEP Move_MM_Step
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
  GOTO Moveit
```

```
Moveit:                                             '*********************************'
FOR Cur_x = Cur_x TO Sensor_Pos STEP 30             'Subroutine that moves servo back '
  PULSOUT Servo_Pos, Cur_x                                 'to the sensor position fast and  '
  PAUSE 20                                          'and then Sensor -> Home slowly    '
NEXT                                                '*********************************'
Cur_x = Cur_x + 30
GOSUB Check_Color
PAUSE 500
GOSUB Home_Position
Drop_mm_Exit:
RETURN

Sensor_Position:                             'Loop to position Servo From HOME -> Sensor
  FOR Cur_x = Home_Pos TO Sensor_Pos STEP 2
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
RETURN

Home_Position:                               'Loop Gets Servo from Current location to Home
  FOR Cur_x = Cur_x TO Home_Pos STEP 4
    PULSOUT Servo_Pos, Cur_x
    PAUSE 20
  NEXT
RETURN

Check_Cal:                                   'Loop to instruct user to calibrate M&M sorter before using
  DEBUG CR, REP "*" \40 , CR
  DEBUG "Please goto Calibration Routine", CR
  DEBUG "And Calibrate this Sorter", CR
  DEBUG REP "*" \40 , CR
END
```

Read EEPROM Program listing for sorter

```
'{$STAMP BS2}
'{$PBASIC 2.5}
```

```
'*********************************************
'     Program = ReadEEPROM Program           '
'     Purpose:  Use this program to          '
'               display the EEPROM data      '
'     Authors:  Rick Johnson & Chris Hileman '
'********************************************'
Word_Var        VAR  Word
BaseR           CON   10
FinishR         CON  5 * 8 + BaseR
RedHigh         VAR   Word
RedLow          VAR   Word
GreenHigh       VAR   Byte
GreenLow        VAR   Byte
BlueHigh        VAR   Byte
BlueLow         VAR   Byte
Pos_Vars        VAR   Word

DEBUG CLS
READ 0, Pos_Vars
DEBUG "Arm - Pos_Delta = ", DEC Pos_Vars, CR
READ 2, Word Pos_Vars
DEBUG "Arm - Hole #1   = ", DEC Pos_Vars, CR
READ 4, Word Pos_Vars
DEBUG "Arm - Sensor    = ", DEC Pos_Vars, CR
READ 6, Pos_Vars
DEBUG "Arm - Home      = ", DEC Pos_Vars, CR
READ 8, Pos_Vars
DEBUG "Cal. Flag       = ", DEC Pos_Vars, CR, CR

FOR Word_Var = BaseR TO FinishR STEP 8
 READ Word_Var, Word RedLow
 READ (Word_Var + 2), Word RedHigh
 READ (Word_Var + 4), GreenLow
 READ (Word_Var + 5), GreenHigh
 READ (Word_Var + 6), BlueLow
 READ (Word_Var + 7), BlueHigh
 DEBUG CR, "RedLow   = ", DEC4 RedLow,  "     RedHigh = ", DEC4 RedHigh, "  Counter = ",DEC3 Word_Var, CR
 DEBUG "GreenLow = ", DEC4 GreenLow, "   GreenHigh = ", DEC4 GreenHigh, CR
 DEBUG "BlueLow  = ", DEC4 BlueLow, "    BlueHigh = ", DEC4 BlueHigh, CR
NEXT
```

```
DEBUG CR, "     Done!"
END
```

<u>Appendix B – Materials List</u>

TCS230 sensor from Texas Advanced Optoelectronic Solutions.
  Company link: www.taosinc.com
  Product link: http://www.taosinc.com/product_detail.asp?cateid=11&proid=12

Sorter hardware available from Parallax, Inc.
  Company link: http://www.parallax.com/
  Sorter hardware: http://www.parallax.com/detail.asp?product_id=30067
  TCS230 sensor: http://www.parallax.com/detail.asp?product_id=30054

Microcontroller and development board from Parallax, Inc.
  Company link: http://www.parallax.com/
  Product link: http://www.parallax.com/detail.asp?product_id=28850
  http://www.parallax.com/detail.asp?product_id=BS2-IC