Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

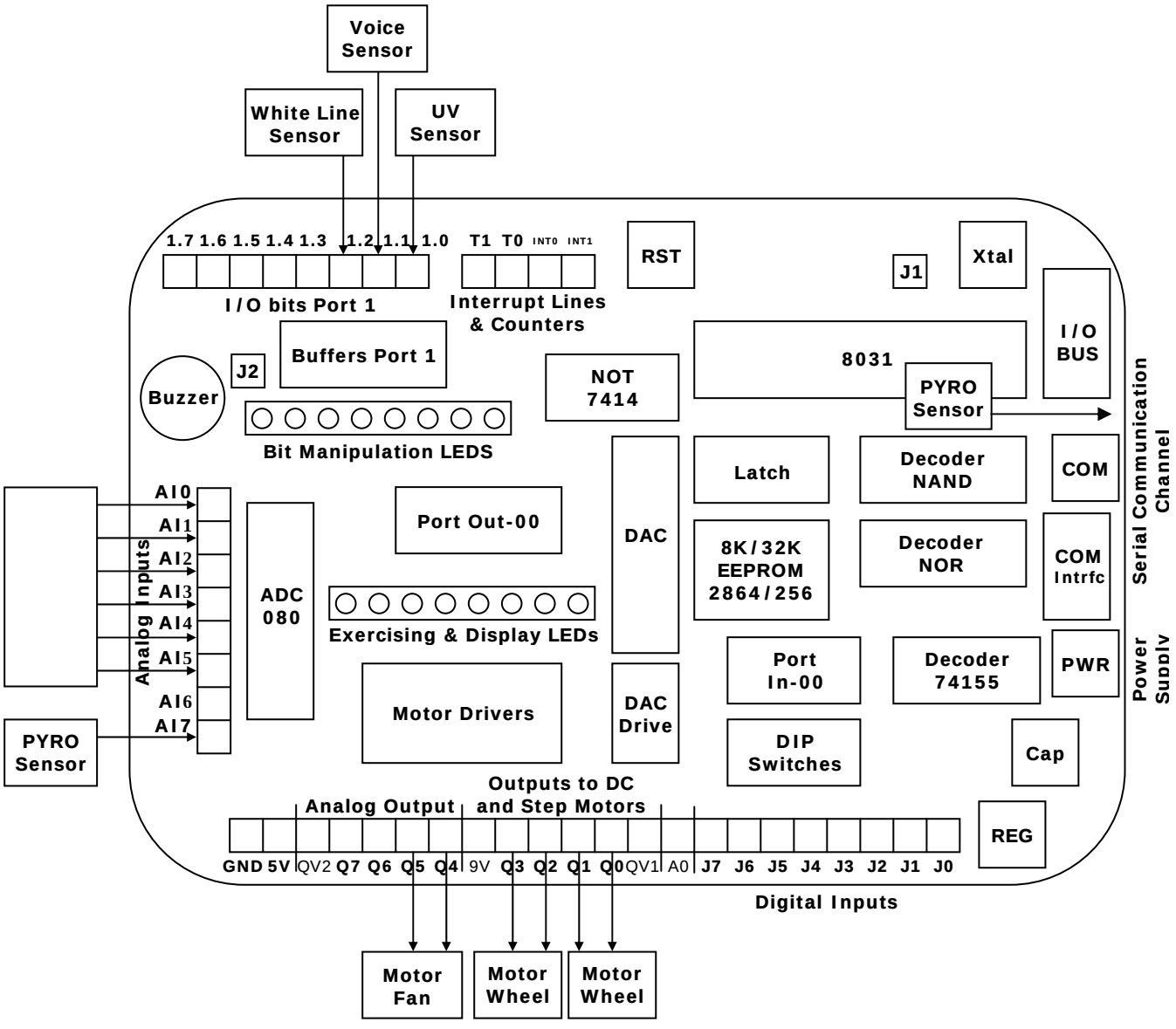
Scientific
Educational
Systems

Electric Connections Diagram for Roboner

A robot for the Roboner exhibition needs, in addition to the robotic card and the structure, the following components:

1) Two DC motors for the wheels.
2) A DC motor for the fan.
3) Six analog distance sensors.
4) A digital white line sensor.
5) A digital voice sensor.
6) A ditigal UV sensor.
7) An analog PYRO sensor.

The following Diagram explains how to connect the above components to the DSM-2095 robotic card:



Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Sensors' using method:

1) DC motors:

Each motor is connected to two driver's outputs (Q0-Q7). In this way, the motor's direction can be controlled. It is advisable to connect each motor to two adjacent outputs: Q0-Q1, Q2-Q3, Q4-Q5, Q6-Q7.

There are two possible ways to control the motor's speed:
The first is by connecting the driver's QV input to the card's DAC output. This method is very simple, but it doesn't allow you to reach the maximal power and it requires that the two motors will always run at the same speed.

The second method is the PWM (Pulse Width Modulation) method. In this method, we supply the motors pulses with variable pulse width that determines the motor's speed. In the "Robotics and Computerized Systems" book, there is an experiment that deals with motors control.

Appendix A describes speed control and operation programs for every motor connected to a pair of outputs (as described above).

Important to remember:

A motor's direction cannot be changed unless first stopped. This change creates a very high voltage on the driver and on the motor. The drivers can contain high voltages and include protection diodes, but they cannot withstand these high voltage pulses.

2) Distance sensor:

The distance sensor is a component fed from a 5V voltage. It releases an analog voltage as a function of its distance from a certain object. The component sends strong Infra-Red light pulses and relates to the returned light pulse.

The sensor is accompanied by a book. It is most advisable to read the book and the usage instructions thoroughly.

- It is most advisable not to connect the distance sensors in parallel to the 5V voltage of the card controller. The component has a very high momentary current consumption. Several sensors together can drop the controller's voltage for a short time and drive the system crazy.
  It is best to supply the sensors with a 5V voltage from a different 5V stabilizer that has a high input capacitor.

Some sensors' output signals are noisy, while others are less. Connecting a 0.1uF filter capacitor reduces the noises substantially. SES recommends connecting a 10uF capacitor in parallel to the sensor's voltage inputs.

The noises ride the signal, thus sampling the signal several times and taking the lower value, can be used as an additional filter. The amount of noise is different from one sensor to another and varies. From a test we made, taking the lower value from 8-12 samples gives quite a stable reading.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Important to remember:

The component is limited to a defined distance. Readings from less than 10 cm range should be avoided because they can be considered bigger then the true value.

Readings from various sensors are not identical. Every sensor should be calibrated separately.

The component is not linear. If linearity is important, a conversion table can be used to convert the readings to linear readings. How to use conversion tables is described in the "Principles of the Microcontroller 8051" guide book.

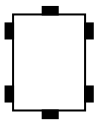
To allow movement along a wall, connect a sensor to each of the robot's side panels. It is advisable to connect even two sensors to each side panel.

It is advisable to transfer the sensors' sample to a timer's interrupt routine that reads each sensor several times and transfers the lower value to a memory cell. The main program relates only to the sensors values located in the memory cells.

Appendix B describes how to read the sensors using interrupt program including finding the lowest value of several readings.

3) Movement control with distance sensors:

In order to fully control the robot's movement, 6 distance sensors are connected around it in the following way:



Straight movement is done by moving the two motors forwards.

A right turn is done by reducing speed from the right wheel or by bringing it to a holt when a sharp turn is needed.

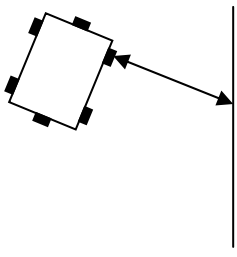
A left turn is done by reducing speed from the left wheel or by bringing it to a holt it when a sharp turn is needed.

An on the spot right rotation is done by moving the left wheel forwards and the right wheel backwards.

An on the spot left rotation is done by moving the right wheel forwards and the left wheel backwards.

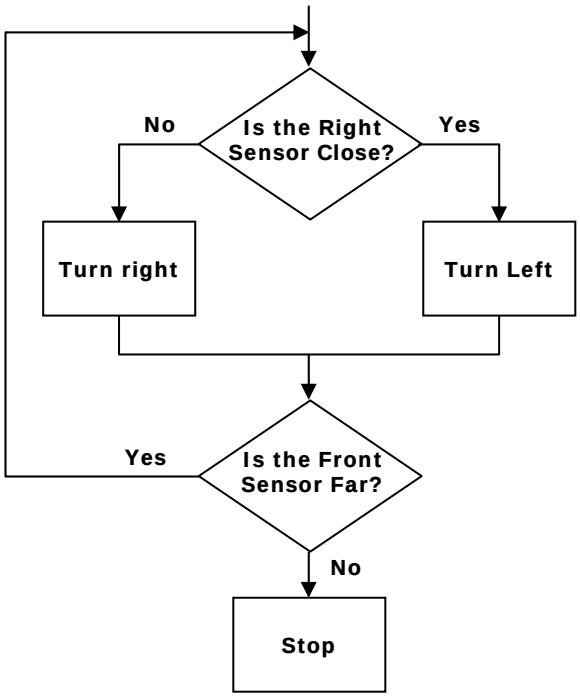
Moving along a wall is done by reading the distance sensor on the side panel and making constant movement decisions.

- Remember that the distance sensor also reads the distance from the wall diagonally. It does not read the projection but the diagonal.



The following program is simple and quite reliable, moving the robot along a right wall. This program performs a right (not sharp) turn when the robot is far from the wall (or even at the desirable distance from the wall), and a left (not sharp) turn when the robot is close to the wall. Actually, the robot constantly performs turns while progressing.

The program is built from the following algorithm:



Appendix C describes this program, which combines working with PWM and reading sensors in the background, including finding the lowest value.

The program can be improved to movement by two sensors.

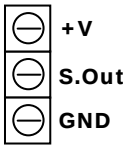
It is advisable to analyze the robot's movement by state simulations and the robot's reaction to those states.

4) Voice sensor:

Operating the robot during the competition is done by bringing the buzzer close to the voice sensor.

The voice sensor (SESPET-81) includes a microphone, an audio amplifier and a connection connector.

The connector has 3 connection points:



A power supply is connected to the +V and GND points. The power supply negative trigger is connected to the GND.

The supply voltage should be 5V. The sensor can be connected to the card's 5V connector. It does not need current and while addressing it, the robot and its systems are not activated.

The S.Out (Signal Out) point is connected to the controller's digital input. It is possible and advisable to connect it to the port 1 input.
If the sensor and the controller are connected to different suppliers, the sensor's GND point should be connected to the controller's GND point.

The signal at the Sout output is a square wave in the sound frequency.

It is most advisable to wrap a small carton cylinder (preferable in the shape of a cone) around the microphone to improve the sensor's sensitivity.

It is recommended to write a program that enumerates the duration of a defined number of ups and downs. As the time is shorter, the frequency is higher.

Another possibility is to count the numbers of ups and downs that appear in a certain counting loop. As the number is higher, the frequency is higher.

Appendix D describes this type of program.

The system should react only to sound above 2000Hz frequency.

The 8051 timers can be used as timers for external signals. The timers inputs appear on the DSM-2095 card and are marked as T0 and T1.

The voice sensor's output can be connected to one of the timers inputs.

The main program initializes the timer to act as a counter. Afterwards, the main program resets the counter, performs a delay loop and reads the counter. If the number is smaller than the desired value, (we have to find a frequency above 2000Hz), the main program repeats the process (reset, delay, read and check). Only when a counting larger then the desired value is received, the program continues.

Appendix D also describes this type of program.

Remember, identification in computerized control is done by checking if a number is bigger or smaller than a certain value and not if equal to a certain value.

Filtering signals in the program is preferable on filtering in the hardware.

5) Buzzer:

The buzzer module (SESPET-02) includes a pushbutton switch, a buzzer and a connector to a 9V battery.

After connecting the 9V battery to the battery plug, pressing the switch causes the buzzer to beep.

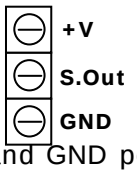
6) White line sensor:

White line sensor (SESPET-82) includes a powerful LED, a light sensor and a connection connector.

The connector has 3 connection points:

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems



A power supply is connected to the +V and GND points. The power supply negative trigger is connected to the GND.

The supply voltage should be 5V.

The S.Out (Signal Out) point is connected to the controller's digital or analog input.

If the sensor and the controller are connected to different suppliers, the sensor's GND point should be connected to the controller's GND point.

When voltage is supplied to the module, the red light appears in the LED.

When the module is directed to the ground, light is reflected back to the sensor. Black background swallows the light and white background reflects the light. The light intensity is particularly strong to reduce the surroundings changing light influences as much as possible. Despite this, the small card should be set under the robot, so that the surroundings' direct light won't reach it. It is preferable to create side panels around it, to block it completley.

SES has two types of white line sensors. The first is a 5x5 black circuit, the second is a smaller green circuit.
The black circuit is an analog sensor (it can also be connected to a digital input). The input voltage is a function of the returned light.
The green circuit also includes a sophisticated enhancement circuit based on two server amplifiers and a poteniometer for sensor's sensitivity adjustment. The circuit releases a digital signal, so it is best to connect it to a digital input.

Both circuits are designed for good readings at the height of 1.5-3 cm above the ground.

Glue a white carton on a black carton. Connect the sensor to the card's 5V triggers and its output to the port 1 input.

Activate the robotic card and press RESET. Move the sensor over the cartons. Check that the green LED reacts to the different backgrounds on the ground. Find the optimal height.

Important to know:

A black background also reflects light. Some reflect quite a lot of light.

The black circuit can also be connected to an analog input.

Write and run a program that reads the analog channel connected to the sensor (see the "Robotics and Computerized Systems" book by SES), and outputs the readings to the LEDs.

Find the binary numbers received over the black background and over the white line according to the distance of the robot's module from the ground. Determine the number that defines the identification of a white line.

Also perform tests with disturbances from surrounding light.

Remember, identification in computerized control is done by checking if a number is bigger or smaller than a certain value and not if equal to a certain value.

7) UV sensor:

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

This sensor includes a UV LED, which should be installed on the controller card.

The sensor is accompanied by a book. It is most advisable to read the book and the usage instruction thoroughly.

Note:

The LED is sensitive to dirt and oiliness, so don't hold it in your hands.

The card can be fed with a stabilized 5V voltage or a direct voltage non-stabilized at the range of 10-30V.

The circuit's bridges should be adjusted according to the book.

The card has a connector with 5 connection points: 3,2,1,-,+.

The voltage is supplied to the + (plus) and – (minus) points accordingly.

Point 1 supplies digital pulse in the CMOS level when a flame radiation is identified.
Point 2 supplies opposite signal to point 1.
Point 3 supplies a signal in the Open collector output.

When a flame is identified, 10ms width pulses are received in the outputs. The pulses can be widened by adding a capacitor at the points marked as CX.

It is advisable not to react to the first pulse, but to count a number of pulses in a short time to make sure that it is indeed a burning flame.

The sensor's output (point 1) is connected to the port 1 input.

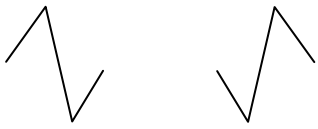
Checking the sensor is done by the commands JB or JNB.

8) PYRO sensor:

This sensor identifies movement of a heat emitting body. It is a sensor that absorbs Infra-Red radiation.

The sensor is accompanied by a book. It is most advisable to read the book and the usage instruction thoroughly.

This sensor outputs an analog signal when the heat emitting body crosses its path. The direction of the crossing determines the signal shape.



Because the candle is immobile, we need to turn the sensor from side to side, until it recognizes the above signal. The signal is received when the sensor is situated opposite to the heat source.

It is preferable to make a double scanning – from right to left and from left to right.

The Acroname company recommends building a carton horn around the sensor using a special lens. This will give a more sharp and clear focus on the heat source location.

Note that how the sensor is located on the card in relation to the robot's movement is important. The sensor has an identifying bulge.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

9) SES' candle sensor

This new sensor is a digital candle sensor with a special structure. It includes three infra – red sensors. Two of the sensors are aligned and are separated by a non-transparent partition. The two sensors locate the light from the candle only when the robot is

standing infront of it. When the robot is standing with an angle to the candle, only one of the sensors can see it.

The third sensor is located undernith the two sensors, designated to abrogate the light coming from the sorroundings. The voltage received from each of the first two sensors is compared to the voltage received from the third sensor. Each sensor has its own potentiometer to colibrate it.

The sensor's module has two digital outputs. These are connected to the controller's digital inputs. Identifing the candle and its location is very simple:
00 – there is no candle around
01 – there is a lit candle left of the module
10 – there is a lit candle right of the module
11 – there is a lit candle infront of the module.

10) Fan:

This unit includes a fan. It's ment for 12V voltage. Different fans can be used.
The fan's device enables installation it on any kind of surface.

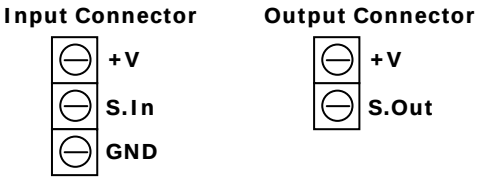
The fan can be connected to the driver's two outputs (Q0-Q7), or between an output to 12V.

11) One-way driver:

This module (SESPET-08) is not a sensor, but a propulsion component.

This module is used when all the DSM-2095 card drivers' outputs are in use and we need to operate another motor or lamp. This module enables turning the motor on and off, but not to control its direction.

The module includes power transistor, input connector and output connector.



The consumer (the motor or the lamp) is connected between the +V and GND output connector's connection points.

The power supply is connected to the +V and GND points in the input connector. The power supply's negative trigger is connected to the GND. The supply voltage should be between 9V and 12V.

The S.In (Signal In) point is connected to one of the DSM-2095 card's port 1 outputs.

Activating this bit is done by the command:

SETB P1.2

This command will activate the consumer.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Resetting this bit is done by the command:

CLR P1.2

This command will disconnect the consumer.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Appendix A – Motors Control

Background program:

The best way to perform motor control with the PWM is with a timer. We activate the timer in the main program and make sure that it will give an interrupt every defined period of time. After performing this procedure, the main program is free to perform the controller's other functions and the timer's interrupt program is working in the background, taking care of the motor's speed control.

The communication between the main program and the timer's interrupt program:

For the main program to instruct the motor's mode of operation (ON, OFF or direction) and the desired speed to the interrupt program, we specify two memory cells, which are used for communication between these two programs.

In one cell, which we name IMAGE (Port Image), the main program will put the number we want to transfer to the output port. The number will determine if the motors will be activated and in what direction.

In the second cell, which we name PWM1 (PWM for motor 1, connected to Q0 and Q1 channels), the main program will put a number in the range of 0-255 (00-FF). This number will determine the motor's ON duration. The higher the number, the longer the duration.

The interrupt program:

The interrupt program uses two additional cells. One cell is called CNTR. This cell increases in one with every interrupt. After it reaches FF and another interrupt is received, it resets.

The interrupt program compares between the number in the PWM1 cell and the number in the CNTR cell. If CNTR is smaller than PWM1, it activates the motor. If CNTR is bigger or equal to PWM1, the interrupt program stops the motor. It will activate the motor again after the CNTR resets and is again smaller than the PWM1.

Program 1:

The following program performs control over one motor, connected to the output channels Q0, Q1.

The main program reads the switches state and stores it in the PWM1 cell. In other words, the state of the switches determines the motor's speed.

The program is assembled of a main program and a Timer0 interrupt program.

The main program starts by initializing the timer and resetting the CNTR. Afterwards the main program can ignore the motor control.

To operate the motor, the main program should write a number in the IMAGE cell. 01 will turn the motor in one direction, 02 will turn the motor in the other direction, and 00 will stop it. The writing is done once each time the motor needs to be turned on or off.

Every time a change is made in the motor's speed, the main program will write a number between 00 and FF in the PWM1 cell. This number will determine the motor's speed.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

```
PWM1    EQU     41H             ;determines Q0,Q1 motor speed
CNTR    EQU     45H
IMAGE   EQU     46H
PORTA   EQU     0FF00H
TINTR   EQU     00H             ;for 10ms timer interrupts
;                               ;increase this number for shorter intervals
        ORG     200BH
        LJMP    TMR0
;
        ORG     2100H
MAIN:   MOV     TMOD,#01H       ;mode1, C/T'=0,GATE=0
        MOV     TH0.#0FFH
        MOV     TL0,#TINTR
        SETB    EA              ;enables all interrupts
        SETB    ET0             ;enables timer0 interrupt
        SETB    TR0             ;starts the timer
        MOV     CNTR,#0
        MOV     DPTR,#PORTA
        MOV     IMAGE,#01       ;motor Q1-Q0 ON
MAI2:   MOVX    A,@DPTR         ;variable speed according to
        MOV     PWM1,A          ;switches to motor1
        SJMP    MAI2
;
TMR0:   CLR     TR0             ;stops the timer
        PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        PUSH    PSW
;
        MOV     DPTR,#PORTA
TMR1:   INC     CNTR
        MOV     A,PWM1
        CLR     C
        SUBB    A,CNTR          ;compare PWM1 with CNTR
        JNC     TMR2            ;jump if CNTR is smaller than PWM1
        MOV     A,IMAGE         ;CNTR is bigger than PWM1
        CLR     ACC.0           ;so it stops the motor
        CLR     ACC.1
        SJMP    TMR3
TMR2:   MOV     A,IMAGE         ;PWM1 is still bigger than CNTR
TMR3:   MOVX    @DPTR,A         ; so it outputs IMAGE
;
        MOV     TH0,#OFFH       ;to create the next timer interrupt
        MOV     TL0,#TINTR
        SETB    TR0
        POP     PSW
        POP     DPL
        POP     DPH
        POP     ACC
        RETI
```

It is important to understand this program, because it is the basics in working with background programs and using a timer for control in real time.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
#include "8052.h"

unsigned char at 0xff00 xdata IOPORT;
unsigned char PWM1;
unsigned char PWM_CNTR;
unsigned char OPORT_IMAGE;
unsigned char TEMP_IMAGE;

unsigned char TINTH=0xFF;
unsigned char TINTR=0;


void tmr0() interrupt 1 using 2
{
 TR0=0;                                          //stops the timer
 TH0=TINTH;                                      //to create the next timer interrupt
 TL0=TINTL;
 TR0=1;                                          //enables the timer

 TEMP_IMAGE = OPORT_IMAGE;
 PWM_CNTR = PWM_CNTR + 4;
 if (PWM_CNTR > PWM)                    //compare PWM counter with PWM1 value
  TEMP_IMAGE = TEMP_IMAGE & 0xfc;       //and stop motor if bigger
 IOPORT = TEMP_IMAGE;
}

void main(void(
{
 TMOD=0x21;              //keep timer1 for comm. and timer0 in mode1
 TH0=TINTH;             //timer0 interval
 TL0=TINTR;
 PWM_CNTR=0;
 ET0=1;
 EA=1;
 TR0=1      ;                                    //start timer0
 OPORT_IMAGE = 01;
 while (1)
 {
  PWM1 = IOPORT;
 }
}
```

The TINTR value determines the time that will pass between interrupts. As we increase this number, we will get shorter times and the motor's movement will be smother. But, 0n the other hand, the CPU will be occupied more with the interrupt program and less in the main program.
The number 00 enables sensing the motor's switches. In actual, it is best to increase it to 80H.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Note:

This program is a bit different from the program in the "Robotics and Computerized Systems" book. The TMR0 initialization is done at the beginning of the interrupt routine and not at its end (as in the book). It allows us to add changes to the routine and keep the interrupt cycle in a unified length. We should make sure that the time needed for executing the interrupt program will not be longer from the time needed for to operate the timer.

A PWM cycle ends every 256 interrupts. In other words, one PWM cycle lasts about 77ms (0.08 second), which equals 12 cycles per second. This is a bit slow (we will feel the motors switching). If we'll want to have a fast movement control, we will get a slow reaction from the system.

Increasing the TINTR value shortens the time between the interrupts, but will allocate less time to the main program. The processor will be busy with the interrupt program for a longer period of time. Instead, a double increasment of the CNTR cell in every interrupt cycle, shortens the PWM cycle by half, without influencing the PWM accuracy. Four enlargements in every interrupt cycle shorten the PWM cycle by fourth.

The increasment is done by adding the INC CNTR command after the line labelled as TMR1 in the program.

The DC motor of Micro Motors, supplied by SES, is designated for a PWM cycle of 50-100 cycles per second. By increasing the CNTR by four, every interrupt cycle will give us the desired pace.

Program 2:

When we want to control more than one motor and enable different speed to each of the motors, we need additional cells. We will call these cells (one cell for each motor): PWM1, PWM2, PWM3, and PWM4.

Motor 1 is connected to channels Q0,Q1; motor 2 to channels Q2,Q3; motor 3 to channels Q4,Q5; motor 4 to channels Q6,Q7.

The interrupt program compares between the CNTR value and each of the cells (PWM1-PWM4) and turns the motor on and off accordingly.

```
PWM1    EQU    41H          ;determines Q0,Q1 motor speed
PWM2    EQU    42H
PWM3    EQU    43H
PWM4    EQU    44H
CNTR    EQU    45H
IMAGE   EQU    46H
PORTA   EQU    0FF00H
TINTR   EQU    00H          ;for 10ms timer interrupts
;                           ;increase this number for shorter intervals
        ORG    200BH
        LJMP   TMR0
;
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

```
        ORG     2100H
MAIN:   MOV     TMOD,#01H       ;mode1, C/T'=0,GATE=0
        MOV     TH0.#0FFH
        MOV     TL0,#TINTR
        SETB    EA              ;enables all interrupts
        SETB    ET0             ;enables timer0 interrupt
        SETB    TR0             ;starts the timer
        MOV     CNTR,#0
        MOV     PWM1,#0FFH      ;maximum speed to motor1
        MOV     PWM2,#0C0H      ;fast speed to motor2
        MOV     PWM3,#80H       ;medium speed to motor3
        MOV     PWM3,#40H       ;slow speed to motor 4
MAI2:   MOV     DPTR,#PORTA
        MOVX    A,@DPTR         ;motors ON according to switches
        MOV     IMAGE,A
        SJMP    MAI2
;
TMR0:   CLR     TR0             ;stops the timer
        PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        PUSH    PSW
;
        MOV     DPTR,#PORTA
        INC     CNTR
        MOV     A,PWM1
        CLR     C
        SUBB    A,CNTR          ;compare PWM1 with CNTR
        JNC     TMR2            ;jump if CNTR is smaller than PWM1
        MOV     A,IMAGE         ;CNTR is bigger than PWM1
        CLR     ACC.0           ;so stop the Q1,Q0 motor
        CLR     ACC.1
        SJMP    TMR3
TMR2:   MOV     A,IMAGE         ;do not change image
TMR3:   MOV     IMAG2,A         ;save image in a temporary cell
;
        MOV     A,PWM2
        CLR     C
        SUBB    A,CNTR          ;compare PWM2 with CNTR
        JNC     TMR4            ;jump if CNTR is smaller than PWM2
        MOV     A,IMAG2         ;CNTR is bigger than PWM2
        CLR     ACC.2           ;so stop the Q3,Q2 motor
        CLR     ACC.3
        MOV     IMAG2,A         ;update imag2
;
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
TMR4:   MOV     A,PWM3
        CLR     C
        SUBB    A,CNTR          ;compare PWM3 with CNTR
        JNC     TMR5            ;jump if CNTR is smaller than PWM3
        MOV     A,IMAG2         ;CNTR is bigger than PWM3
        CLR     ACC.4           ;so stop the Q5,Q4 motor
        CLR     ACC.5
        MOV     IMAG2,A         ;update imag2
;
TMR5:   MOV     A,PWM4
        CLR     C
        SUBB    A,CNTR          ;compare PWM4 with CNTR
        JNC     TMR6            ;jump if CNTR is smaller than PWM4
        MOV     A,IMAG2         ;CNTR is bigger than PWM4
        CLR     ACC.6           ;so stop the Q7,Q6 motor
        CLR     ACC.7
        MOV     IMAG2,A         ;update imag2
TNR6:   MOV     A,IMAG2
        MOVX    @DPTR,A         ;output imag2
;
        MOV     TH0,#0FFH       ;to create the next timer interrupt
        MOV     TL0,#TINTR
        SETB    TR0
        POP     PSW
        POP     DPL
        POP     DPH
        POP     ACC
        RETI
```

In order to not influence the number that the main program outputs to the port (IMAGE), we use an additional memory cell, IMAG2.

The main program writes a number in the IMAGE cell according to the motors it wishes to activate. This data is given to the IMAG2 cell at the beginning of each interrupt cycle. For example, the number 85H (10000101) activates the motors Q0,Q1 and Q2,Q3 in one direction, stops the motor Q4,Q5, and turn the motor Q6,Q7 in the other direction.

An interrupt cycle starts when cell CNTR is zeroed. This cell increases in one with every interrupt. It counts from 00 to FF and so forth.

When the CNTR is zeroed, we transfer the IMAGE content to IMAG2. Every time the interrupt program finds that one of the motors needs to be turned off, it zeros the bits connected to this motor.

At the end of the interrupt routine (after the comparisons), the program outputs the IMAG2 content to the output port.

To illustrate this, the program sets maximum speed for motor 1, high speed for motor 2, medium speed for motor 3, and low speed for motor 4.

Determing which motors will run and in which direction is doen by the switches.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

A C language four motors control program:

```
#include "8052.h"
unsigned char at 0xff00 xdata OPORT;
unsigned char PWM1;
unsigned char PWM2;
unsigned char PWM3;
unsigned char PWM4;
unsigned char PWM_CNTR;
unsigned char OPORT_IMAGE;
unsigned char TEMP_IMAGE;

unsigned char TINTH=0xFF;
unsigned char TINTR=0;


void tmr0() interrupt 1 using 2
{
 TR0=0;                                          //stops the timer
 TH0=TINTH;                                        //to create the next timer interrupt
 TL0=TINTR;
 TR0=1;                                          //enables the timer

 TEMP_IMAGE = OPORT_IMAGE;
 PWM_CNTR = PWM_CNTR + 4;
 if (PWM_CNTR > PWM1)                    // compare PWM counter with PWM1 value
  TEMP_IMAGE = TEMP_IMAGE & 0xfc;        //and stop motor if bigger
 if (PWM_CNTR > PWM2)                    //compare PWM counter with PWM2 value
  TEMP_IMAGE = TEMP_IMAGE & 0xf3;        //and stop motor if bigger
 if (PWM_CNTR > PWM3)                    //compare PWM counter with PWM3 value
  TEMP_IMAGE = TEMP_IMAGE & 0xcf;        //and stop motor if bigger
 if (PWM_CNTR > PWM4)                    //compare PWM counter with PWM4 value
  TEMP_IMAGE = TEMP_IMAGE & 0x3f;        //and stop motor if bigger

 OPORT = TEMP_IMAGE;
}

void main(void)
{
 TMOD=0x21;              //keep timer1 for comm. and timer0 in mode1
 TH0=TINTH;             //timer0 interval
 TL0=TINTL;
 PWM_CNTR=0;
 ET0=1;
 EA=1;
 TR0=1     ;                                     //start timer0

 PWM1 = 255;
 PWM2 = 128;
 PWM3 = 255;
 PWM4 = 32;
 while (1)
  {
  OPORT_IMAGE = OPORT;
  }
}
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Appendix B – Reading Distance Sensors in the Background

To release the main program from addressing the distance sensors, we'll use, once again, a timer. In every timer's interrupt, we sample one sensor and put it into a memory cell. Certainly, it is possible to sample all the sensors, but then the interrupt program will be too long and it will influence the main program's speed.

The method we sample by a different sensor each time allows us to concede the waiting for the ECO (End Of Conversion) signal.

At each interrupt, we read one channel and create SOC for the next channel. The time between the interrupts should be longer than the conversion's time.

The following program samples the eight ADC channels and stores them in the eight memory cells (ADC0-ADC7).

The main program outputs the state of the AI0 channel to port FF00 and the AI7 channel state to port 1.

Program 3:

```
ADC0    EQU   40H
ADC1    EQU   41H
ADC2    EQU   42H
ADC3    EQU   43H
ADC4    EQU   44H
ADC5    EQU   45H
ADC6    EQU   46H
ADC7    EQU   47H
INDEX   EQU   48H
PORT0   EQU   0F00H
PAI0    EQU   0FF08H
TINTR   EQU   00H          ;for 10ms timer interrupts
;                          ;increase this number for shorter intervals
        ORG   200BH
        LJMP  TMR0
;
        ORG   2100H
ADC:    MOV   TMOD,#01H    ;mode1, C/T'=0,GATE=0
        MOV   TH0.#0FFH
        MOV   TL0,#TINTR
        SETB  EA           ;enables all interrupts
        SETB  ET0          ;enables timer0 interrupt
        SETB  TR0          ;starts the timer
        MOV   INDEX,#0
        MOV   DPTR,#PAI0   ;creates SOC for ADC0
        MOVX  @DPTR,A
ADC2:   MOV   A,ADC0
        MOV   DPTR,#PORT0
        MOVX  @DPTR,A
        MOV   P1,ADC7
        SJMP  ADC2
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
;
TMR0:   CLR     TR0             ;stops the timer
        PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        PUSH    PSW
;
        MOV     R0,#ADC0
        MOV     DPTR,#PAI0
        MOV     A,INDEX
        CJNE    A,#0,TMR2
        SJMP    TMR3
TMR2:   INC     DPTR
        INC     R0
        DJNZ    ACC,TMR2
TRM3:   MOVX    A,@DPTR         ;reads analog channel
        MOV     @R0,A           ;and saves it
        INC     DPTR
        INC     INDEX
        MOV     A,INDEX
        CJNE    A,#8,TMR4
        MOV     DPTR,#PAI0
        MOV     INDEX,#0
TMR4:   MOVX    @DPTR,A         ;creates SOC for next channel
        MOV     TH0,#0FFH       ;to generate the next timer interrupt
        MOV     TL0,#TINTR
        SETB    TR0
        POP     PSW
        POP     DPL
        POP     DPH
        POP     ACC
        RETI
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

A C language background sample of 8 analog channels:

```
#include "8052.h"
unsigned char at 0xff00 xdata OPORT;
unsigned char TINTH=0xFF;
unsigned char TINTL=0;
unsigned char TINTR=0;

unsigned char at 0xff08 xdata ADC0;
unsigned char at 0xff09 xdata ADC1;
unsigned char at 0xff0a xdata ADC2;
unsigned char at 0xff0b xdata ADC3;
unsigned char at 0xff0c xdata ADC4;
unsigned char at 0xff0d xdata ADC5;
unsigned char at 0xff0e xdata ADC6;
unsigned char at 0xff0f xdata ADC7;
unsigned char ANALOG_IN0;
unsigned char ANALOG_IN1;
unsigned char ANALOG_IN2;
unsigned char ANALOG_IN3;
unsigned char ANALOG_IN4;
unsigned char ANALOG_IN5;
unsigned char ANALOG_IN6;
unsigned char ANALOG_IN7;
unsigned char INDEX;

void tmr0() interrupt 1 using 2
{
 TR0=0;                                          //stops the timer
 TH0=TINTH;                                      //to create the next timer interrupt
 TL0=TINTL;
 TR0=1;                                          //enables the timer

 switch (INDEX)
 {
 case 0:
              ANALOG_IN0 = ADC0;          //read channel 0
              ADC1 = 0;                   //create soc for channel 1
              INDEX = 1;
              break;
 case 1:
              ANALOG_IN1 = ADC1;          //read channel 1
              ADC2 = 0;                   //create soc for channel 2
              INDEX = 2;
              break;
 case 2:
              ANALOG_IN2 = ADC2;          //read channel 2
              ADC3 = 0;                   //create soc for channel 3
              INDEX = 3;
              break;
 case 3:
              ANALOG_IN3 = ADC3;          //read channel 3
```

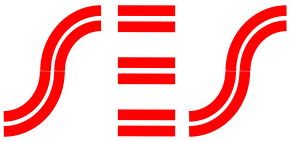
Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
                ADC4 = 0;                       //create soc for channel 4
                INDEX = 4;
                break;
 case 4:
                ANALOG_IN4 = ADC4;  //read channel 4
                ADC5 = 0;               //create soc for channel 5
                INDEX = 5;
                break;
 case 5:
                ANALOG_IN5 = ADC5;  //read channel 5
                ADC6 = 0;               //create soc for channel 6
                INDEX = 6;
                break;
 case 6:
                ANALOG_IN6 = ADC6;  //read channel 6
                ADC7 = 0;               //create soc for channel 7
                INDEX = 7;
                break;
 case 7:
                ANALOG_IN7 = ADC7;  //read channel 7
                ADC0 = 0;               //create soc for channel 0
                INDEX = 0;
                break;
 }
}

void main()
{
 TMOD=0x21;              //keep timer1 for comm. and timer0 in mode1
 TH0=TINTH;              //timer0 interval
 TL0=TINTL;
 ET0=1;
 EA=1;
 TR0=1     ;                     //start timer0
 INDEX = 0;

 while (1)
 {
 OPORT = ANALOG_IN0;
 P1 = ANALOG_IN7;
 }
}
```

The following program executes eight samples at each sensor and updates the sensor's cell with the lowest value of the eight samples.

The program uses additional counter named SAMPLE, which is calibrated at the initialization stage in the main program and a cell named TEMP for temporary storage of the lowest reading during the samples. We place FF in this cell every time we start a new set of samples.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il



Scientific
Educational
Systems

Program 4:

```
ADC0    EQU   40H
ADC1    EQU   ADC0+1
ADC2    EQU   ADC1+1
ADC3    EQU   ADC2+1
ADC4    EQU   ADC3+1
ADC5    EQU   ADC4+1
ADC6    EQU   ADC5+1
ADC7    EQU   ADC6+1
INDEX   EQU   ADC7+1
SAMPLE  EQU   INDEX+1
TEMP    EQU   SAMPLE+1
;
PORT0   EQU   0FF00H
PAI0    EQU   0FF08H
TINTR   EQU   00H           ;for 10ms timer interrupts
;                           ;increase this number for shorter intervals
        ORG   200BH
        LJMP  TMR0
;
        ORG   2100H
ADC:    MOV   TMOD,#01H     ;mode1, C/T'=0,GATE=0
        MOV   TH0.#0FFH
        MOV   TL0,#TINTR
        SETB  EA            ;enables all interrupts
        SETB  ET0           ;enables timer0 interrupt
        SETB  TR0           ;starts the timer
        MOV   INDEX,#0
        MOV   SAMPLE,#0
        MOV   TEMP,#0FFH
        MOV   DPTR,#PAI0    ;creates SOC for ADC0
        MOVX  @DPTR,A
ADC2:   MOV   A,ADC0        ;read analog channel 0
        MOV   DPTR,#PORT0
        MOVX  @DPTR,A
        MOV   P1,ADC7
        SJMP  ADC2
;
TMR0:   CLR   TR0           ;stops the timer
        MOV   TH0,#0FFH     ;to generate the next timer interrupt
        MOV   TL0,#TINTR
        SETB  TR0
        PUSH  ACC
        PUSH  DPH
        PUSH  DPL
        PUSH  PSW
;
        MOV   R0,#ADC0
        MOV   DPTR,#PAI0
        MOV   A,INDEX
        CJNE  A,#0,TMR2
        SJMP  TMR3
TMR2:   INC   DPTR
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
        INC     R0
```

```
        DJNZ    ACC,TMR2
TMR3:   MOVX    A,@DPTR         ;read analog channel
        MOV     B,A
        CLR     C
        SUBB    A,TEMP          ;compare with TEMP
        JC      TMR4            ;if lower than
        MOV     TEMP,B          ;replace TEMP with the new number
TMR4:   INC     SAMPLE
        MOV     A,SAMPLE
        CJNE    A,#8,TMR5
        MOV     A,TEMP          ;after 8 samples
        MOV     @R0,A           ;save it in its cell
        MOV     SAMPLE,#0
        MOV     TEMP,#0FFH
        INC     DPTR            ;next channel
        INC     INDEX
        MOV     A,INDEX
        CJNE    A,#8,TMR5
        MOV     DPTR,#PAI0      ;next channel is AI0
        MOV     INDEX,#0
TMR5:   MOVX    @DPTR,A         ;creates SOC for next sample
        POP     PSW
        POP     DPL
        POP     DPH
        POP     ACC
        RETI
```

A C language background sample of eight analog channels, saving the lowest value:

```
include "8052.h"
unsigned char at 0xff00 xdata OPORT;
unsigned char TINTH=0xFF;
unsigned char TINTL=0;
unsigned char TINTR=0;

unsigned char at 0xff08 xdata ADC0;
unsigned char at 0xff09 xdata ADC1;
unsigned char at 0xff0a xdata ADC2;
unsigned char at 0xff0b xdata ADC3;
unsigned char at 0xff0c xdata ADC4;
unsigned char at 0xff0d xdata ADC5;
unsigned char at 0xff0e xdata ADC6;
unsigned char at 0xff0f xdata ADC7;
unsigned char ANALOG_IN0;
unsigned char ANALOG_IN1;
unsigned char ANALOG_IN2;
unsigned char ANALOG_IN3;
unsigned char ANALOG_IN4;
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
unsigned char ANALOG_IN5;
```

```
unsigned char ANALOG_IN6;
unsigned char ANALOG_IN7;
unsigned char INDEX;
unsigned char SAMPLE;
unsigned char MIN;
unsigned char TEMP

void tmr0() interrupt 1 using 2
{
 TR0=0;                                          //stops the timer
 TH0=TINTH;                                      //to create the next timer interrupt
 TL0=TINTL;
 TR0=1;                                          //enables the timer

 switch (INDEX)
 {
 case 0:
              TEMP = ADC0;
              if TEMP < MIN
                MIN = TEMP;
              SAMPLE++;
              if SAMPLE > 7
               {
                ANALOG_IN0 = MIN;
                MIN = 0xFF;
                SAMPLE = 0;
                ADC1 = 0;                    //create soc for channel 1
                INDEX = 1;
               }
              break;
 case 1:
              TEMP = ADC1;
              if TEMP < MIN
                MIN = TEMP;
              SAMPLE++;
              if SAMPLE > 7
               {
                ANALOG_IN1 = MIN;
                MIN = 0xFF;
                SAMPLE = 0;
                ADC2 = 0;                    //create soc for channel 2
                INDEX = 2;
               }
              break;
case 2:
              TEMP = ADC2;
              if TEMP < MIN
                MIN = TEMP;
              SAMPLE++;
              if SAMPLE > 7
               {
                ANALOG_IN2 = MIN;
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
                MIN = 0xFF;
```

```
                  SAMPLE = 0;
                  ADC3 = 0;                    //create soc for channel 3
                  INDEX = 3;
                }
                 break;
case 3:
               TEMP = ADC3;
               if TEMP < MIN
                  MIN = TEMP;
               SAMPLE++;
               if SAMPLE > 7
                {
                  ANALOG_IN3 = MIN;
                  MIN = 0xFF;
                  SAMPLE = 0;
                  ADC4 = 0;                    //create soc for channel 4
                  INDEX = 4;
                }
                break;
case 4:
               TEMP = ADC4;
               if TEMP < MIN
                  MIN = TEMP;
               SAMPLE++;
               if SAMPLE > 7
                {
                  ANALOG_IN4 = MIN;
                  MIN = 0xFF;
                  SAMPLE = 0;
                  ADC5 = 0;                    //create soc for channel 5
                  INDEX = 5;
                }
                break;
case 5:
               TEMP = ADC5;
               if TEMP < MIN
                  MIN = TEMP;
               SAMPLE++;
               if SAMPLE > 7
                {
                  ANALOG_IN5 = MIN;
                  MIN = 0xFF;
                  SAMPLE = 0;
                  ADC6 = 0;                    //create soc for channel 6
                  INDEX = 6;
                }
                break;
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
 case 6:
              TEMP = ADC6;
              if TEMP < MIN
                MIN = TEMP;
              SAMPLE++;
              if SAMPLE > 7
               {
                ANALOG_IN6 = MIN;
                MIN = 0xFF;
                SAMPLE = 0;
                ADC7 = 0;                          //create soc for channel 7
                INDEX = 7;
               }
               break;
 case 7:
              TEMP = ADC7;
              if TEMP < MIN
                MIN = TEMP;
              SAMPLE++;
              if SAMPLE > 7
               {
                ANALOG_IN7 = MIN;
                MIN = 0xFF;
                SAMPLE = 0;
                ADC0 = 0;                          //create soc for channel 0
                INDEX = 0;
              }
               break;
 }
}

void main()
{
 TMOD=0x21;              //keep timer1 for comm. and timer0 in mode1
 TH0=TINTH;              //timer0 interval
 TL0=TINTL;
 ET0=1;
 EA=1;
 TR0=1     ;                          //start timer0
 INDEX = 0;
 SAMPLE = 0;
 MIN = 0xFF;

 while (1)
 {
 OPORT = ANALOG_IN0;
 P1 = ANALOG_IN7;
 }
}
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Appendix C – Movement Control with Distance Sensors

The following program moves the robot along a right wall according to the algorithm described in section 3. The program assumes that the right motor is connected to Q0,Q1 and the left motor is connected to Q2,Q3. When so the number 5 (00000101) moves the robot forwards. The turns are done by changing the motors' speed.

The program assumes that a frontal sensor is connected to the AI0 channel and the right sensor is connected to the AI1 channel.

The desirable distance from the right wall is received when the reading is 30H. The stopping distance from the front wall is received when the reading is 40H.

Note:

The bigger the number, the closer the sensor is to the wall.

Program 5:

```
PWM1    EQU   41H          ;determines Q1,Q0 motor speed – right motor
PWM2    EQU   PWM1+1       ;determines Q3,Q2 motor speed – left motor
PWM3    EQU   PWM2+1       ;determines Q5,Q4 motor speed
PWM4    EQU   PWM3+1       ;determines Q7,Q6 motor speed
CNTR    EQU   PWM4+1
IMAGE   EQU   CNTR+1
ADC0    EQU   IMAGE+1
ADC1    EQU   ADC0+1
ADC2    EQU   ADC1+1
ADC3    EQU   ADC2+1
ADC4    EQU   ADC3+1
ADC5    EQU   ADC4+1
ADC6    EQU   ADC5+1
ADC7    EQU   ADC6+1
INDEX   EQU   ADC7+1
SAMPLE  EQU   INDEX+1
TEMP    EQU   SAMPLE+1
;
PORT0   EQU   0FF00H
PAI0    EQU   0FF08H
TINTR   EQU   00H          ;for 0.3ms timer interrupts
;
        ORG   200BH
        LJMP  TMR0
;
        ORG   2100H
MAIN:   MOV   TMOD,#01H    ;mode1, C/T'=0,GATE=0
        MOV   TH0.#0FFH
        MOV   TL0,#TINTR
        SETB  EA           ;enables all interrupts
        SETB  ET0          ;enables timer0 interrupt
        SETB  TR0          ;starts the timer
        MOV   CNTR,#0
        MOV   IMAGE,#05H   ;Q3-Q2, Q1-Q0 motors ON
        MOV   INDEX,#0
        MOV   SAMPLE,#0
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
        MOV     TEMP,#0FFH
        MOV     DPTR,#PAI0      ;creates SOC for ADC0
        MOVX    @DPTR,A
MAI2:   MOV     A,ADC1          ;read right sensor
        CLR     C
        SUBB    A,#30H          ;compare with the required distance
        JC      MAI3            ;if close or equal turn left, if far turn right
        MOV     PWM2,#0FFH      ;far - high speed to motor2
        MOV     PWM1,#0A0H      ;slow speed to motor1
        LJMP    MAI4
MAI3:   MOV     PWM2,#0A0H      ;close or equal - slow speed to motor2
        MOV     PWM1,#0FFH      ;high speed to motor1
MAI4:   MOV     A,ADC0          ;read front sensor
        CLR     C
        SUBB    A,#40H          ;compare with the required distance
        JNC     MAI2            ;if far than do another loop
        MOV     IMAGE,#0        ;if close than stop
MAI5:   LJMP    MAI5            ;wait for reset
;
TMR0:   CLR     TR0             ;stop the timer
        MOV     TH0,#0FFH       ;update it
        MOV     TL0,#TINTR
        SETB    TR0             ;enable the timer for the next timer interrupt
        PUSH    ACC
        PUSH    DPH
        PUSH    DPL
        PUSH    PSW
;
        MOV     DPTR,#PORT0
TMR1:   INC     CNTR
        INC     CNTR
        INC     CNTR
        INC     CNTR
        MOV     A,PWM1
        CLR     C
        SUBB    A,CNTR          ;compare PWM1 with CNTR
        JNC     TMR2            ;jump if CNTR is smaller than PWM1
        MOV     A,IMAGE         ;CNTR is bigger than PWM1
        CLR     ACC.0           ;so stop the Q1,Q0 motor
        CLR     ACC.1
        SJMP    TMR3
TMR2:   MOV     A,IMAGE         ;do not change image
TMR3:   MOV     IMAG2,A         ;save image in a temporary cell
;
        MOV     A,PWM2
        CLR     C
        SUBB    A,CNTR          ;compare PWM2 with CNTR
        JNC     TMR4            ;jump if CNTR is smaller than PWM2
        MOV     A,IMAG2         ;CNTR is bigger than PWM2
        CLR     ACC.2           ;so stop the Q3,Q2 motor
        CLR     ACC.3
        MOV     IMAG2,A         ;update imag2
;
TMR4:   MOV     A,PWM3
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il


```
        CLR     C
        SUBB    A,CNTR          ;compare PWM3 with CNTR
        JNC     TMR5            ;jump if CNTR is smaller than PWM3
        MOV     A,IMAG2         ;CNTR is bigger than PWM3
        CLR     ACC.4           ;so stop the Q5,Q4 motor
        CLR     ACC.5
        MOV     IMAG2,A         ;update imag2
;
TMR5:   MOV     A,PWM4
        CLR     C
        SUBB    A,CNTR          ;compare PWM4 with CNTR
        JNC     TMR6            ;jump if CNTR is smaller than PWM4
        MOV     A,IMAG2         ;CNTR is bigger than PWM4
        CLR     ACC.6           ;so stop the Q7,Q6 motor
        CLR     ACC.7
        MOV     IMAG2,A         ;update imag2
TNR6:   MOV     A,IMAG2
        MOVX    @DPTR,A         ;output imag2
;
        MOV     R0,#ADC0
        MOV     DPTR,#PAI0
        MOV     A,INDEX
        CJNE    A,#0,TMR2
        SJMP    TMR8
TMR7:   INC     DPTR            ;adapt DPTR
        INC     R0              ;and R0 to INDEX
        DJNZ    ACC,TMR7
TMR8:   MOVX    A,@DPTR         ;read analog channel
        MOV     B,A
        CLR     C
        SUBB    A,TEMP          ;compare with TEMP
        JC      TMR9            ;if lower than
        MOV     TEMP,B          ;replace TEMP with the new number
TMR9:   INC     SAMPLE
        MOV     A,SAMPLE
        CJNE    A,#8,TMR10
        MOV     A,TEMP          ;after 8 samples
        MOV     @R0,A           ;save it in its cell
        MOV     SAMPLE,#0
        MOV     TEMP,#0FFH
        INC     DPTR            ;next channel
        INC     INDEX
        MOV     A,INDEX
        CJNE    A,#8,TMR10
        MOV     DPTR,#PAI0      ;next channel is AI0
        MOV     INDEX,#0
TMR10:  MOVX    @DPTR,A         ;creates SOC for next sample
;
        POP     PSW
        POP     DPL
        POP     DPH
        POP     ACC
        RETI
```

- Check this program. You may find typing mistakes.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Appendix D – Reaction to Voice Sensor and Waiting for Sound

Program 6:

This program performs a waiting for the first pulse. Afterwards it executes FFFF delay loops while counting the receiving pulses. The number of pulses received is stored in the accumulator A.

```
SNDB    EQU     p1.0            ;assuming that the sensor output is connected to p1.0
plss    EQU     40              ;minimum number of pulses to be counted in an interval of 0.1 second
;
        ORG     2100H
SND:    MOV     R7,#0FFH
        MOV     R6,#0FFH        ;to create 0.1 second delay
        MOV     A,#0            ;pulse counter
        CLR     F0              ;checking flag
        JNB     SNDB,SND        ;wait for the first pulse
;
SND2:   JB      SNDB,SND3       ;wait for pulse dropping
        JNB     F0,SND4         ;don't count if flag=0
        INC     A               ;count if F0=1 and SNDB=0
        CLR     F0
        LJMP    SND4
SND3:   SETB    FO              ;when SNDB=1, raise flag
SND4:   DJNZ    R6,SND2
        MOV     R6,#OFFH
        DJNZ    R7,SND2
        CLR     C
        SUBB    A,#PLSS
        JC      SND
;
;       The program continue from here
;
        END
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Program 7:

This program sets TIMER0 as a counter, under the assumption that the voice sensor's output is connected to the timer's input.

The program zeroes the counter and then executes a delay loop. At the end of the delay loop, it checks the number in the counter. If the number is bigger than the desired value, it means that the correct sound has been received. It executes this process again for safety. If the number is smaller than the desired value, it executes the process again.

```
                ORG     2000H
                LJMP    MAIN
;
                ORG     2100H
MAIN:           MOV     A,#255
                MOV     P1,A            ;to indicate status
;
                CLR     ET0             ;disable TIMER0 interrupt
                MOV     TMOD,#25H       ;TIMER0 16 bit counter from T0 pin
                                        ;TIMER1 in mode 2 to enable to return to the debugger
;
WAIT_BUZ:
                CLR     TR0
                CLR     TF0
                MOV     TH0,#OFFH
                MOV     TL0,#255-100    ;to count 155 pulses until overflow
                SETB    TR0             ;enable counter

                LCALL   DLY_80MS
                JNB     TF0,WAIT_BUZ    ;buzzer will cause counter set T1 flag
;
                CLR     TR0             ;once again
                CLR     TF0
                MOV     TH0,#OFFH
                MOV     TL0,#255-100    ;to count 155 pulses until overflow
                SETB    TR0             ;enable counter

                LCALL   DLY_80MS
                JNB     TF0,WAIT_BUZ    ;buzzer will cause counter set T1 flag
;
                MOV     A,#55H          ;to indicate receiving the pitch
                MOV     P1,A
                CLR     TR0
                LCALL   103H            ;return to the debugger

DLY_80MS:
                MOV     R7,#144
DLY2:           MOV     R6,#0FFH
DLY3:           DJNZ    R6,DLY3
                DJNZ    R7,DLY2
                RET
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

The same program in C language:

```
#include "8052.h"

int I;

void main(void)
{
  P1 = 0xFF;
  ET0 = 0;
 TMOD=0x25;
 do
 {
   TR0 = 0;
   TF0 = 0;
   TH0 = 0xFF;
   TL0 = 255-100;
   TR0 = 1;
   for (I = 0; I < 1000; I++);
 }
 while (TF0 = 0);
 P1 = 0x55;
 TR0 = 0;

while (1)
 {
}
}
```

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Planning Ahead

Our natural desire is to to create and implement. Detailed planning is conceived as boring and as an annoying demand. Many times, we begin building a complicated system with some initial idea and a very basic sketch. For those who start this way, it is guaranteed that the developing process will be long, accompanied by many changes and bitter disappointments.

Investing in a detailed planning before starting to work may seem like the long way, but it is always the fastest way. For those of you who wish to walk this way, I wrote the following notes:

1) Write the product's characterization and sketching for yourself and your friends. The act of writing creates organized thinking and organized analyzing of the product. A discussion with a written document also becomes organized, clear, and comprehensive.
2) Work with the attitude of "from the system to the components". If you do not plan correctly, you will find that some components have no room on the circuit, or that the system is not stable etc.
3) Write the product's characterization in a Word Processor in order to edit and improve it all the time. You can make extra copies for your team members.
4) Get used to convert written remarks to typed text.
5) Dismantle the characterization to: product's definition, goals and performances, components, structure characterization, hardware characterization and software characterization.
6) Use short numbered sentences in a graded form. For example:

    1. Product's name
    2. Product's description
    3. Goals and performances

    3.1 ....
    3.2 ....

    4. Components

    4.1 ....
    4.2 ....

    5. Structure

    5.1 ....
    5.2 ....

    6. Hardware

    6.1 ....
    6.2 ....

    7. Software

    7.1 ....
    7.2 ....

7) Draw a block diagram of the system and its components.
8) Draw a block diagram of the software. Designing an embedded micro-computer system requires solutions that combine software and hardware.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

9) Changes are much harder to do than a systematical construction of the system. As you invest more time and thinking in the characterization, you'll need less time for changes during the ectual work.
10) Do not start building or programming before the whole picture of the product and its components is clear to you. This means the characterization and not the complete parts; otherwise, you will never start working.
11) In a teamwork, each member receives a task. If the components characterization is explicit and understood, the integration stage will be simpler.
12) Run an organized file with all the aid sheets, drafts (do not throw the drafts, they will remind you your way of thinking), remarks and discussions summations, data sheets, brochures, software lists, prices etc. This file will accompany you until you finish the project.
13) Divide this file to organized sections.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Structure and Mechanics

1) Design and draw all the mechanical parts.
2) Write down the list of components and check if you have them before starting to build. Make sure you are not left with a screwdriver and no screw.
3) Search and use existing means. Avoid, as much as possible, from "inventing the wheel". Despite this, do not assume that everything was already invented. You may come up with a good innovative idea. Search propulsion component and transmission.
4) Remember – simple is reliable, nice and strong.
5) Use the appropriate tools.
6) **Safety above all**. Keep the safety rules.
7) Even a strict design is not without errors. Sometimes a new need is created or ideas for change or improvement come up. Thus, always think that you are going to manufacture a series of products even if you plan to manufacture only one product.
8) Prepare an organized drawing for each part.
9) Prepare a cutting template for the production material.
10) Prepare a drilling template for each small (1-2mm) drilling series at the suitable places. Next to each hole, write the hole diameter, which will be in the real material. Before each drilling in the real material, drill a little hole with the template's help.
11) Build a model from a soft material before using the real one, in order to see how the product will look and to predict manufacturing difficulties. A carton is an excellent mean for templates and models.
12) Do not depend on your memory. Write every remark, idea and measurement on the drawings and templates. **"The opposite from remembrance is writing"**. If you write more and try to remember less, you will make fewer mistakes.
13) Believe me, working with models and templates shortens the manufacturing time. It also allows to manufacture a damaged part very fast, not to mention duplicating the product.
14) After all those measurments and markings, drink a glass of water and measure again. It is extremely frustrating to find a hole or a bend that is not accurate.
15) An easy material for structure is foamed PVC. This material is quite hard, but can be easily cut and bend.
16) After building the model, use it to mark the cutting lines on the PVC sheet. Cut the material carefully.
17) Mark the bending lines on the material.
18) Grab the material along the bending line with a vice. If the vice is too short, use metal rulers or aluminum profiles to adjust them to the bending lines.
19) Heat the material from both sides along the bending line. Use an electric hair drier or a heating device.
20) When you feel that the material is soft and bendable, bend it with any wood sheet. The wood sheet will protect your hand and will create a unified and straight bend. Keep pressing on the material until it cools.
21) Remember that a certain bending radius is created in the bending line.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

Hardware

1) Make sure that the electronic drawing of the product is clear to you. The electronic people in your team must understand the operational mode of the electronic circuit, so they can locate errors in the circuit itself and its components.
2) The mechanic and programming people do not have to understand the electronic circuit, but it's important they'll understand its components, how to address them, the data passing through them, their parameters and performances.
3) It is important to understand the function of each connection point to the external components and its limitation (voltage, current etc.).
4) Before building the circuit, make a list of all the circuit's components and check their compatibility to the drawing and to their location onthe circuit. Check that you have all the required components.
5) Use an intact soldering iron, its base and a wet sponge for cleaning the soldering iron during welding.
6) Make sure that your work table is clean and tidy and that you have enough space.
7) Use electronics tin, which contains cleaning flax. This tin will improve the quality of your welding. The flax draws out dirt and oiliness located at the welding point.

Software

1) Finish the software characterization and the performance definitions before writing the program. Software written in patches causes bugs and a terrible waste of time.
2) Build the main program in such way that it will use the sub programs.
3) An organized program starts with the basic routines and only than the main program. Our aim is that every routine will read and use the routines above her (which are already defined, written and checked).
4) The main program can be written in parts, so it will be used to check the basic routines.
5) Write to yourself the development and checking order. This way, you will be able to hold the software development schedule.
6) Define the basic help routines used by the main program (forward, backward, right, left, search etc.). These routines are called BIOS (Basic Input Output Subroutines).
7) Write a short characterization of every routine – **name, performances, the variables it uses, the values it brings back, the help routines it uses**.
8) The program and routines documentation are designate for your use only, and has no formal rule.
9) If you'll trust your memory less, the development will run faster.
10) Build each routine do each check separately. Never write a whole program and then check it as one piece.
11) Prepare a checking procedure for each routine. Remember, every bug in the program increases the debugging time in exponential and not in a linear. The checking procedure causes us to think about each routine as a separate program and to debug it in a reliable way.
12) Sometimes, during the development process, we need to change a previous routine. Making a change, even if it seems quite simple, creates new bugs. If we have a written checking procedure, we do not need to rely on our memory. Checking the routine after making the change will be reliable and fast.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

Scientific
Educational
Systems

DSM-2095 Software

The monitor program includes various routines that are used by it. The card user can also use these routines as help routines. To make it easier on the user and on the documentation, addressing tables for the various routines were written at the beginning of the Debugger program. Addressing the help routine is done by accessing an address in the addressing table. A skip command to the required routine is written in the table. This way, even if the location of the routine has changed in update versions, the routine portal address is fixed.

The DSM-2095 communication with the computer is done with a communication component called UART, located inside the processor. The card's monitor set it in the required work mode. The component uses TIMER1 and MODE2.

Before using the following routines, make sure the program does not change the TIMER1 mode. TMOD should receive the number 2X (see program 6 as an example).

Every routine is a sub-routine ending with the RET command (return from the sub-routine).

The On Line Debugger sub-routines:

a) WARM - ("warm" entrance) to the Debugger – 0103:

Calling this routine while running programs under the Debugger, causes the program to return to the Debugger, which print a point on the screen and waits for the user's command. This call is written at the end of the program and it is used as an END sentence. This routine stores the various registers. With the return to the Debugger the command R<ENTER> can be given to display the registers on the screen.

We can use this routine (LCALL 0103) when we need a breaking point in the program and we don't want to do a running with a breaking point, that slows the CPU's work down.

b) CCI – Receiving a character from the keyboard – 0106:

This routine waits for a key pressing on the keyboard. When a key is pressed, a return to the main program is executed. The pressed key ASCII code will appear in the accumulator. The routine does not transmit an echo to the terminal. The program only changes the ACC.

c) CCO – Printing a character on the screen – 0109:

This routine prints a character, which its ASCII code is in the ACC, on the display. After the printing, a return to main program is executed only if the ^S key is not pressed. If the ^S key is pressed, the program waits for the ^@ pressing.

The routine's input data is the ACC.

The program does not change every register.

d) INKEY – Receiving a character from the terminal without waiting – 010C:

This routine checks if a character was received from the terminal. If not, a return to the main program is executed when the number FF is in the accumulator. If a character was received, it will appear in the ACC. The program changes only the ACC.

e) TEXT – Printing text in the terminal – 010F:

This routine prints in the terminal a text in ASCII code located in the Program area. The text line must end with the 00 byte. The text start address should be in the DPTR before calling the routine. The program changes the ACC and the DPTR. The program calls the CCO.

Arazim Building, 20a Eliau Eithan St.,
New Industrial Area, Rishon-Lezion
P.O.Box 5340, Rishon-Lezion 75151
Tel: 972-3-9412457/9, Fax: 972-3-9412425
E-mail: sesltd@netvision.net.il

f) INLINE – Receiving a line from the terminal – 0112:

This routine receives a line of characters from the keyboard. When the <ENTER> key is pressed, a return to main program is executed. The collection of the typed characters will be found in the Buffer, which is located in the external RAM starting with the address BFE0H. The routine also responds to the <BS> (Back Space) key.

The program changes the ACC and DPTR registers.

The program calls the CCI and CCO routines.

g) BIN2ASC – Converting a binary number to 2 digits in ASCII code – 0115:

This routine disassembles a binary number, located in the accumulator, to 2 digits and translate them to ASCII. The translated digits will be found in the R2 and R3 registers. This routine enables the transmission of a binary number to the terminal. The routine input data is the ACC – the number for conversion.

The output data is in R2 and R3. R2 will contain the more significant bit and R3 will contain the less significant bit. The program changes the ACC, R2, R3, and R4 registers.

h) ASC2BIN – Converting 2 digits in ASCII code to a binary number:

This routine converts 2 digits in ASCII code to a binary number, and unifies them into one binary number. The 2 digits for conversion should be located in the R2 and R3 registers. The conversion result will be in the accumulator.

The routine input data is in R2 and R3. R2 will contain the more significant bit and R3 will contain the less significant bit.

The routine output data is in the ACC, which includes the conversion result. The program changes only the accumulator.

i) AOUT – Printing the accumulator content in the terminal – 011B:

This routine converts the number in the accumulator into 2 digits in ASCII code and transfers them to the terminal one after the other; First, the more significant bit.

The routine input data is in the accumulator, which includes the transferred number.

The program changes the ACC, R2, R3, and R4 registers.

The program calls the BIN2ASC and the CCO.