Designing and Implementing Information Display Consoles for Lecturers

by

Ahmad Rizal bin Muhammad Arif

FINAL PROJECT REPORT

Submitted to
Electrical & Electronics Engineering Department
in partial fulfillment of the requirement for the
Bachelor of Engineering (Hons)
(Electrical & Electronics Engineering)

DECEMBER 2004

Universiti Teknologi PETRONAS
Bandar Seri Iskandar
31750 Tronoh
Perak Darul Ridzuan

CERTIFICATION OF APPROVAL

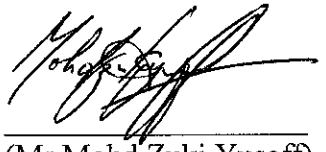
Designing and Implementing
Information Display Consoles for Lecturers

by

Ahmad Rizal bin Muhammad Arif

A project dissertation submitted to the
Electrical & Electronics Programme
Universiti Teknologi PETRONAS
in partial fulfillment of the requirement for the
BACHELOR OF ENGINEERING (Hons)
(ELECTRICAL & ELECTRONICS ENGINEERING)

Approved by,



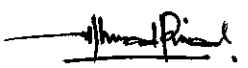
(Mr Mohd Zuki Yusoff)
Project Supervisor

UNIVERSITI TEKNOLOGI PETRONAS
BANDAR SERI ISKANDAR
31750 TRONOH
PERAK

December 2004

CERTIFICATION OF ORIGINALITY

This is to certify that I am responsible for the work submitted in this project, that the original work is my own except as specified in the references and acknowledgements, and that the original work contained herein have not been undertaken or done by unspecified sources or persons.



(AHMAD RIZAL BIN MUHAMMAD ARIF)
820810-05-5323
Matric ID:1922

ABSTRACT

This report is about an Information Display Console (IDC) project that is to be used by each lecturer in Universiti Teknologi PETRONAS (UTP). The purpose of this IDC is to give information to students or lecturers while the particular lecturer is away and to enhance the professionalism in communication among lecturers and students.

The system consists of two parts. Firstly is the transmitter which is connected to a computer via parallel port. Second part is the receiver to which a Liquid Crystal Display (LCD) is attached. The system uses the infrared communication in transmitting data that need to be updated. The data will be stored in a memory and will be displayed continuously. The users can use a computer with its user interface to update the new data. The development of the project involves designing and implementing the hardware and the software. In the hardware part, it discusses the circuit design including the calculation. In the software part, it discusses the process of developing the graphical user interface (GUI), which is to be used in the system. The discussion on testing and debugging the system are also included.

The system is a pioneer system in UTP since it uses new design in hardware and software development. With this successful prototype development, it is envisioned that lectures can enhance their professionalism in communication. Perhaps the system could be deployed in UTP widely.

ACKNOWLEDGEMENTS

Special gratitude to the Final Year Project Committee for managing the final year project paper and also for their responsibilities to ensure that a project would be completed and delivered within the project time frame.

Firstly, I would like to thank to my supervisor, Mr. Mohd Zuki Yusoff for his time providing resources and giving direction for me in completing the project. I am also profoundly grateful to the lab assistant, Miss Hawa and all lab technicians for being very helpful in clarifying certain theories and helping me throughout the duration. My special tribute to Mr Mohd Zuki Yusoff for his encouragements, advises and the priceless feedbacks, which drive and motivate me while completing the thesis.

My deepest thanks also go to all lecturers and staff who support and help me during the duration. I also would like to thank to all, whose names are not mentioned here, for their supports and cooperation in any form. Besides, I also would like to thank my family members and relatives for their moral supports, concerns and understandings. Last but not least, to those who assist me directly or indirectly in making this project successful. Thank you.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS AND NOMENCLATURES

IDC:	Information Display Console
I^2C:	Inter–Integrated Circuit
LCD:	Liquid Crystal Display
LED:	Light Emitting Diode
MFC:	Microsoft Foundation Class
PC:	Personal Computer
SCL:	Clock Signal Line (Serial CLock)
SDA:	Data Signal Line (Serial DAta)
SIRCS:	SONY Infra Red Coding System

CHAPTER 1

INTRODUCTION

The Information Display Console (IDC) will display several items such as the lecturer's name, extension number, consultation hours (Day/Time), class session (Day/Time/Venue), test conducted (Day/Date/Time/Venue), and away message (where the lecturer can be reached if he/she is away). The console will be located at each lecturer's room. Thus, students or anybody can easily know where to find the lecturer if the lecturer is away even if it is during the consultation hours. The students also can know that their lecturer is having a class or has to be away from office.

The IDC will display the stored data to the Liquid Crystal Display (LCD) continuously. The data will be updated only when there is a transmission of data to the IDC. To update the data, the users need to transmit the data from the Personal Computer (PC).

1.1 Background of Study

For this project there are several knowledge areas to be mastered. These are:

1. The principles of the micro controller
2. Inter-Integrated Circuit (I^2C) Protocol
3. Parallel protocol
4. Infrared protocol
5. Microsoft Foundation Class (Visual C++) programming
6. Basic circuit theories

1.2 Problem Statement

There is a problem in which students want to meet a lecturer but the lecturer is not around even, perhaps, during his/her consultation hours. The students might wonder why the lecturer is not around, whether the lecturer is having a class or has an emergency case. Thus, with this IDC the lecturer could put a message on it telling others that he/she is not around and leave some message. Besides, the lecturer can put on his/her schedule time or any relevance items. Also, the IDC can reduce the use of paper.

1.3 Objectives and Scope of Study

The objectives of the project are:

1. Give information to students or lecturers while the particular lecturer is away
2. To enhance the professionalism in communication among lecturers and students
3. To minimize the use of paper

The scope that has to be covered is:

1. PIC16F871/PIC16F84 assembly language programming
2. LCD protocol
3. I^2C protocol
4. Infrared protocol
5. Basic circuit theories
6. Parallel protocol
7. Microsoft Foundation Class (Visual C++) programming

CHAPTER 2

LITERATURE REVIEW

In completing the project, some application on LCD and infrared have been referred from the internet [1], [4] & [5]. While the knowledge of programming the micro controller, interfacing the Non-Volatile Memory (NVM), interfacing the LCD, infrared protocol (SONY Infra Red Coding System = SIRCS) and developing software using Microsoft Foundation Class (MFC) were acquired during the industrial training in SONY Technology (M) Sdn. Bhd. Bangi. While doing the training, one project (LCC Project) has been performed that dealt with PIC16F871 and the NVM. Based on this experience, the knowledge is adapted to this project. Many of the assembly language source code regarding interfacing the NVM and the LCD were developed during the internship program under the supervision of the plant supervisor. The source code used in this project is derived from those codes.

The following are the theory of Inter-Integrated Circuit (I^2C) protocol, LCD protocol, Infrared, Parallel port and microcontroller.

2.1 Inter-Integrated Circuit

Inter-Integrated Circuit (I^2C) protocol was originally developed as a control bus for linking microcontroller and peripheral ICs for Phillips consumer product. It is two wire bus combining address and data bus. The bus data rate is 100kbps [2].

I^2C Features:

a. Resistant to glitches and noise
b. Supported by a large and diverse range of peripheral devices
c. A well-known robust protocol

d. A long track record in the field
e. A respectable communication distance
f. Compatibility with a number of processor with integrated I^2C port
g. Easily emulated in software by any microcontroller
h. Available from a number of component manufacturer

I^2C Terminology

a. Transmitter
b. Receiver
c. Master
d. Slave
e. SDA – Data Signal Line (Serial DAta)
f. SCL – Clock Signal Line (Serial CLock)

Terminology for bus transfer

a. F (FREE) – the bus is free, SDA and SCL are in HIGH state
b. S (START) – data transfer begins with a start condition. The SDA is going from HIGH to LOW, while the SCL remains HIGH
c. C (CHANGE) – while the SCL is LOW, the data bit to be transferred can be applied to the SDA data line be a transmitter. During this time, SDA may change it's state as long as the SCL line remains LOW
d. D (DATA) – a HIGH or LOW bit of information on the SDA line data line is valid during the HIGH level of the SCL clock line. This level must be maintained stable during the entire time that the clock remains HIGH to avoid misinterpretation as a START and STOP condition
e. P (STOP) – data transfer is terminated by a STOP condition (not a stop bit). This occurs when the level on the SDA Data line passes from LOW state to HIGH state, while the SCL clock line remains HIGH. When the data transfer has been terminated, the bus is free once again

Figure 2.1 shows the I^2C bus protocol [3].

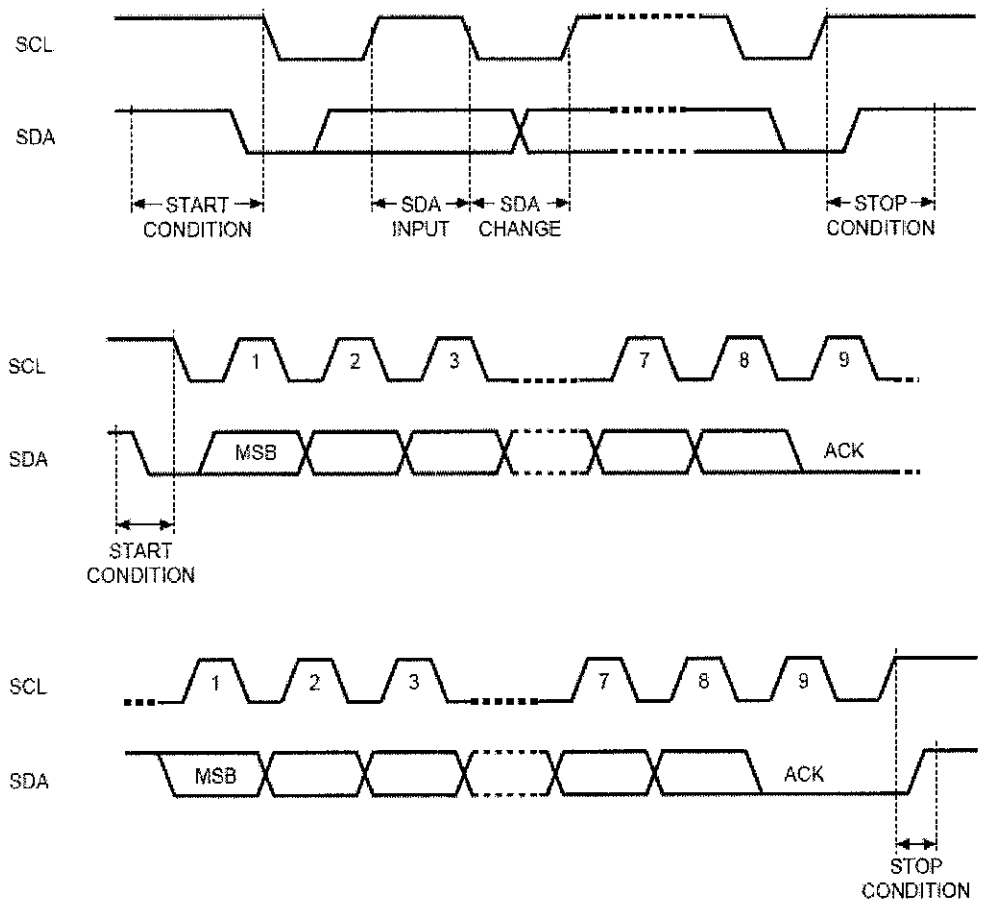


FIGURE 2.1: I^2C Bus Protocol

I^2C Operations [3]

a. Write Operation

Master sends a START condition to the slave. Following the START condition the master sends device select code with the R/*W bit reset to '0'. The slave then acknowledges this and waits for a byte address. Then the master sends the byte address and followed by acknowledged by slave. This sequence is repeated for multibyte write. Then the master sends STOP condition to terminate the communication. Figure 2.2 shows the write mode sequence.

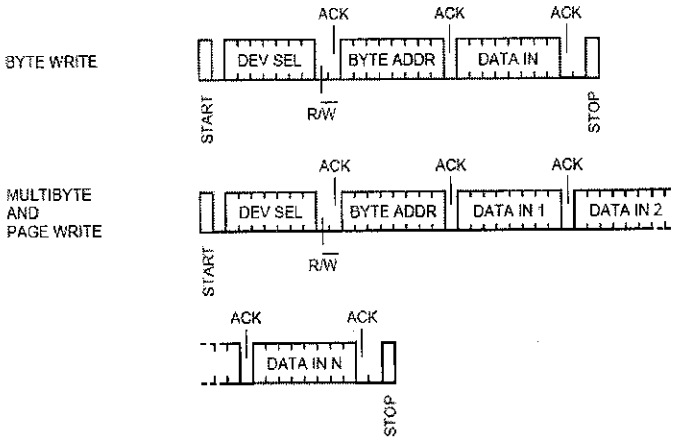


FIGURE 2.2: Write Mode Sequence

b. Read Operation

The read operation is as shown in Figure 2.3.

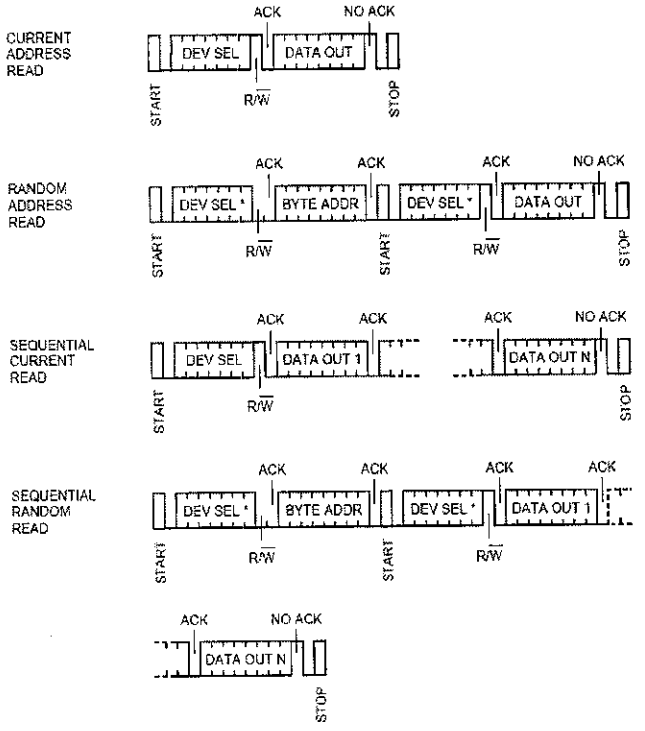


FIGURE 2.3: Read Mode Sequence

2.2 Infrared

When adapting the remote control system on equipment, the following shall be obeyed in order to actualize intercontrol between products and to prevent malfunction in between Sony products, other companies' equipment by the Sony commander, or malfunction of Sony's remote commander by the other company's remote commander

1. Generally, SIRCS shall be used for remote control system
2. If it is functions, the characteristics of the products cannot be actualized, system other that SIRCS can be adopted. In this case, confirmation of the following shall be made:
    a. There shall be no malfunction between its system and products of SIRCS system, products with remote control system in Sony adopted formerly, or remote control in other company
    b. Regulations, ministerial ordinance, etc, shall be satisfied
3. Products supplied by OEM from other companies should adopt the SIRCS system. Sony's products must be operated by Sony's uni-commander (system commander). Other company's commander should not operate Sony's product.
4. When a Sony's product is supplied to other company's product with its brand, SIRCS system should not be used as far as possible. If any problem arise using SIRCS system, it should be dealt by their responsibility

Guide Pulse and Bit Format

The guide pulse and bit format of the remote control signal shall be as follows (12-bit format)

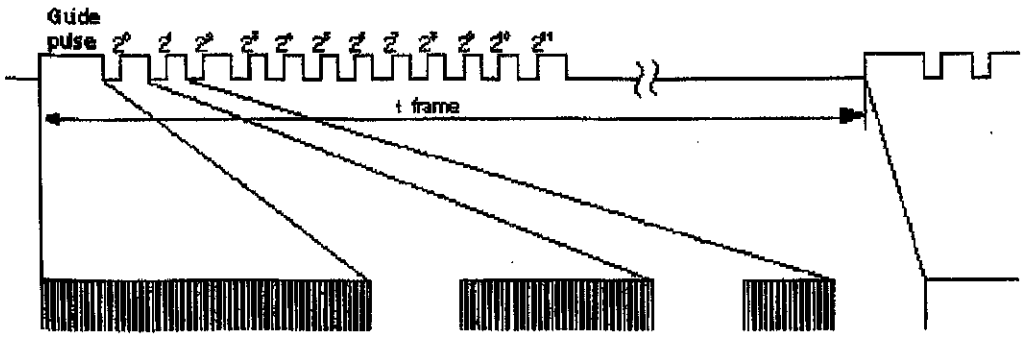


FIGURE 2.4: Infrared Modulated Signal

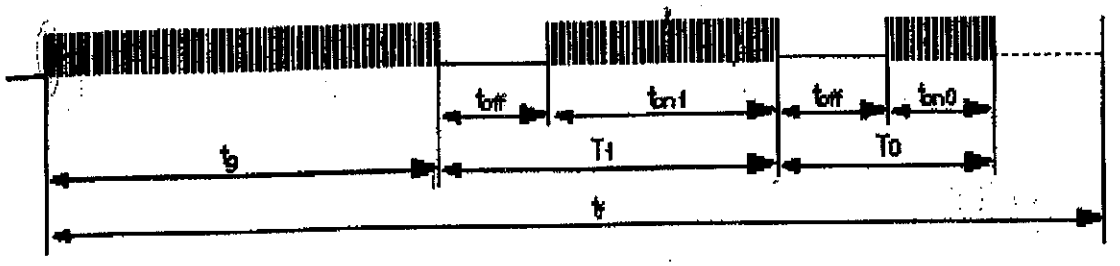


FIGURE 2.5: Timing Diagram

TABLE 2.1: Time duration for each single bit

		Symbol	Time (ms)	Tolerance (ms)
Duration of Guidepulse		t_g	2.4	+/- 0.015
Data bit OFF time		t_{off}	0.6	+/- 0.015
Data bit ON time	"1"	t_{on1}	1.2	+/- 0.015
	"0"	t_{on0}	0.6	+/- 0.015
Width of data bit	"1"	T1	1.8	+/- 0.03
	"0"	T0	1.2	+/- 0.03
Frame period		t_f	45.0	+/- 1.2

Figure 2.6, illustrates the SIRCS protocol [5]. This protocol is standard protocol used by Sony's remote control system. Base on this concept, some modification has been made

where the device code is eliminated. Thus, the protocol left the start pulse and the command code.

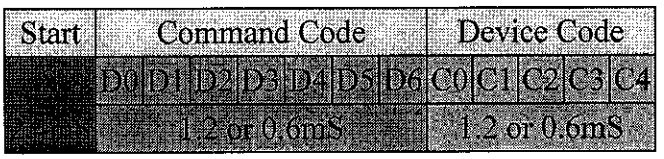


FIGURE 2.6: SIRCS Protocol

2.3 Parallel Port

The original IBM-PC's Parallel Printer Port had a total of 12 digital outputs and 5 digital inputs accessed via 3 consecutive 8-bit ports in the processor's I/O space [7].

- 8 output pins accessed via the **DATA Port**
- 5 input pins (one inverted) accessed via the **STATUS Port**
- 4 output pins (three inverted) accessed via the **CONTROL Port**
- The remaining 8 pins are grounded

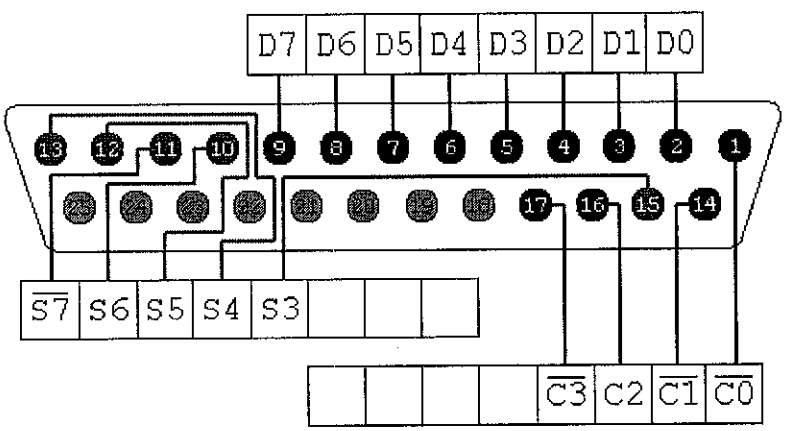


FIGURE 2.7: Parallel Pinout

2.4 Microcontroller

The microcontrollers used are PIC16F871 and PIC16F84. The features of this device can be found in datasheet [10 & 11].

2.5 Encoding and Decoding Infrared

The Sony Infra Red Coding System (SIRCS) was referred back to recap the pulse width modulation system that is being used.

Thus, Figure 2.4 shows how the data bit '1' and '0' is modulated with 38 kHz carrier. The ON state duration is showed in Table 1. The guidepulse is assumed as startpulse and it has 2.4mS ON state, data bit '1' has 1.2mS ON state and data bit '0' has 0.6mS ON state. The width of data bit '1' is 1.8mS and data bit '0' is 1.2mS as shown in Figure 2.5.

Note that the output from the filter is in HIGH state and it goes LOW state when there is an input. Thus, the filter will outputs such that: data bit '1' will has LOW state for a duration of 1.2mS, data bit '0' will has LOW state for duration of 0.6mS as shown in Figure 2.8. Same goes to the startpulse where it has LOW state for duration of 2.4mS.

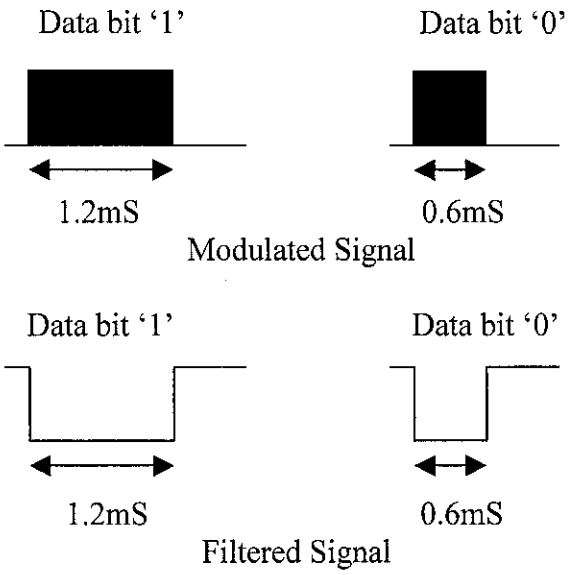


Figure 2.8: Modulated and Filtered Signal for data bit '1' and '0'

Theoretically, the input and the output should appear as shown in Figure 2.8. Practically, the input for the filter (modulated signal) can be generated as shown. The problem rise when it comes to the output of the filter which, it has a delay of some millisecond. Thus, the calculation in determination of data bit '1' and '0' cannot take the theory value. It must be modified in such that it takes into account the delay time.

The calculation for determination of data bit '1' and '0' is based on the transition from HIGH to LOW and from LOW to HIGH of the filter output. The idea is that, the microcontroller starts counting when there is a transition from HIGH to LOW. It stops counting when there is a transition from LOW to HIGH. If the calculated value is less than or equal to the theory value, thus the microcontroller will decode the transmitted data accordingly.

Let's take the example of determination of data bit '0'. It has 0.6mS LOW state duration. First, it has to determine the theory value of 0.6ms (in decimal). Let say the microcontroller will start count when the transition occur (HIGH to LOW). It will keep counting with increase by one for every 10uS until there is a transition from LOW to HIGH. Final value will be 60 decimal: (0.6mS/10uS = 60), data bit '1' (1.2mS/10uS = 120), startpulse (2.4mS/10uS = 240). These are the theory value. Since the output of the

filter having some millisecond of delay, the theory value has been modified such that data bit '0' takes 80 decimal, data bit '1'takes 130 decimal, and startpulse takes 240 decimal as a tolerance of the filter output.

Back to the determination of data bit '0'. Now it has modified theory value of 80 decimal. When data bit '0' appear in the filter output, the microcontroller start to count. The final value then will be compared to the theory value. If it has less or equal to 80, then the microcontroller will decode the signal as data bit '0'. This routine same goes to determination of data bit '1' and startpulse. One feature has been added which is an error detection signal. The error signal is defined as having the value of less than 60 decimal. This, feature is added since there is another source of infra red from the atmosphere. Thus with this feature, it helps to prevent the microcontroller from response to the infra red signal from atmosphere.

CHAPTER 3

METHODOLOGY

There are two processes, which need to be taken in completing the projects. Those processes include procedure identification and tool identification. This process is important to make sure the project is completed smoothly. Procedure identification includes the steps need to be done in completion of the project while tool identification lists all the tools needed in this project.

3.1 Procedure Identification

Steps involved:

1. **Identifying, searching and gathering materials involve**
    a. Acquiring datasheets
        i. PIC16F871
        ii. PIC16F84
        iii. ST24C08 – Serial 8K (1K x 8) EEPROM
        iv. Transistor (2N3904 & 2N3906)
        v. LCD User Manual
        vi. IS1U60/IS1U60L – Sensor with 1-Package Design of Remote Control Detecting Function

    b. Acquiring source code
        i. LCD Functions Code
        ii. I^2C Functions Code

2. **Understanding the material (datasheets)**

   i. PIC16F871 – microcontroller descriptions, pin descriptions, function of register, instruction code used, electrical characteristics

   ii. PIC16F84 - microcontroller descriptions, pin descriptions, function of register, instruction code used, electrical characteristics

   iii. ST24C08 – description, pin descriptions, device operation, electrical characteristics

   iv. 2N3904/2N3906 – pin description, electrical characteristics

   v. LCD User Manual – pin descriptions, instruction sets, electrical characteristics

3. **Identifying, designing, and implementing the hardware needed (receiver and transmitter)**

   The hardware designed and implemented were the receiver, transmitter with parallel receptacle.

   i. Receiver

      The receiver part consists of PIC16F871 micro controller, the 24C08R memory, LCD display, and IR receiver. The interface was based on the electrical characteristics of respective components such as the micro controller, 24C08R memory and the LCD. The characteristics are, the V_{OH}, V_{IH}, V_{OL},V_{IL}, I_{IH}, I_{IL}, I_{OH}, I_{OL}, for every components mentioned before. Since all the characteristics are met by each component, therefore the components are connected directly. The schematic circuit is shown in Figure 3.1.

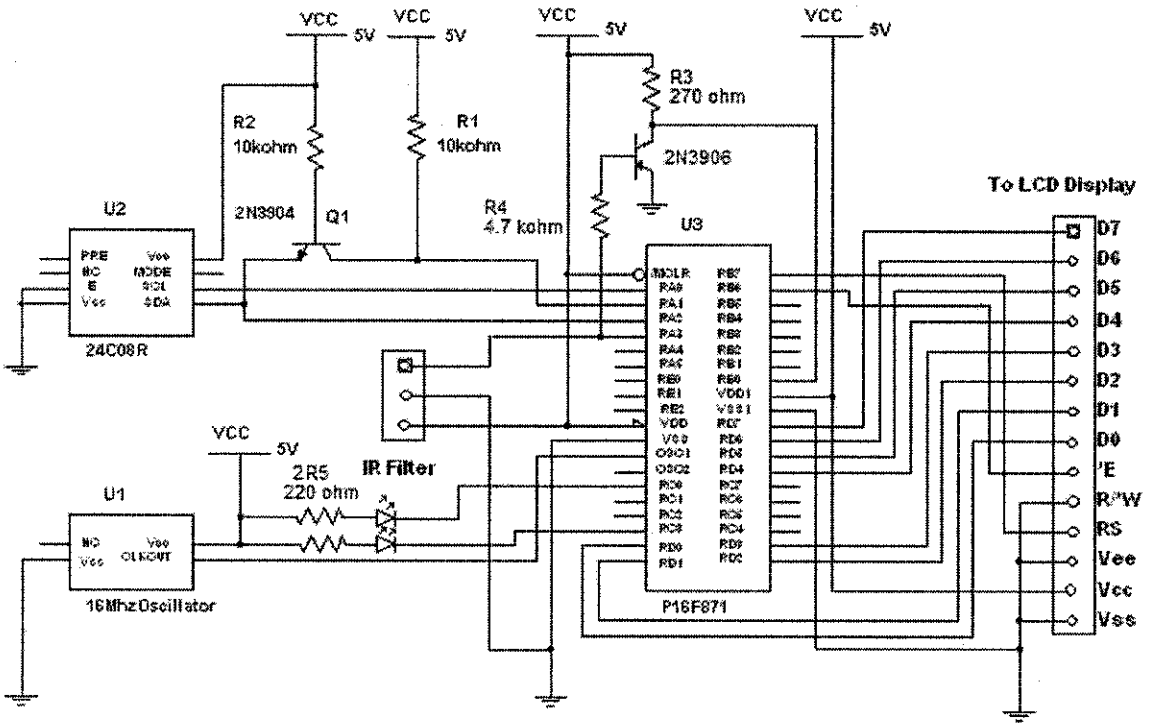


Figure 3.1: Receiver Schematic

ii. Transmitter

The transmitter with parallel receptacle consists of buffer (74LS245N), PIC16F84 microcontroller, parallel receptacle and infra red LED.

The determination of connection for this circuit is the same as the receiver which is based on the electrical characteristics of the components

For the switching ON and OFF of the infra red LED, extra circuit is needed

LED characteristics:

$V_{P(max)} = 1.7$ V

$V_{R(max)} = 5$V

$I_{F(max)} = 100$ mA

2N3904 transistor characteristics

ON characteristics

$H_{fe} = 30$; $I_C = 100$ mA; $V_{CE} = 1.0$V

Resistance value, R_B and R_C

Calculation:

$$5 - V_{CE} - I_C R_C = 0$$

$$5 - 1.0V - (100mA)R_C = 0$$

$$R_C = 40\ \Omega$$

$$I_B = I_C / h_{fe} = 100mA / 30$$

$$V_{in} - V_{BE} - I_B R_B = 0$$

$$4.3\ V - 0.7V - (3.3333mA)R_B = 0$$

$$R_B = 1080\Omega \approx 1k\Omega$$

The schematic circuit is shown in Figure 3.2.

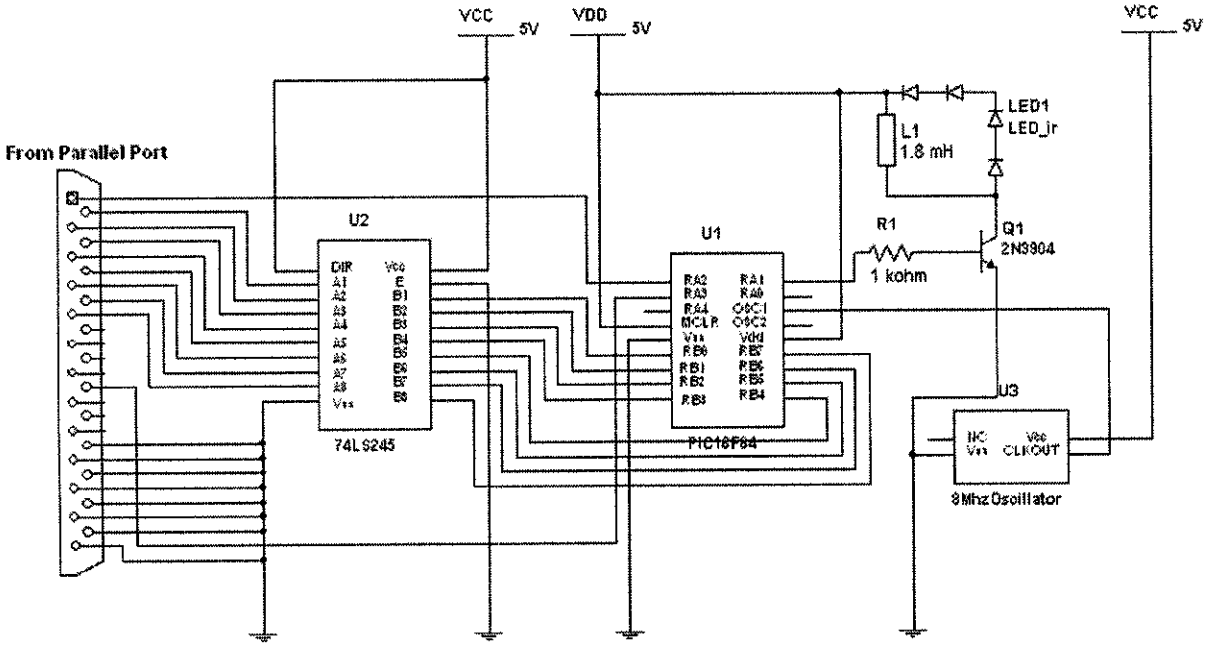


Figure 3.2: Transmitter (parallel) Schematic

4. Program the PIC16F871 – displaying character, read from and write onto memory

The source code that has been developed will be displaying character/sentence. The character to be displayed is obtained from a memory. This microcontroller must display the character read from a memory continuously. At the same time it must be able to handle an interrupt when the users want to update any information in the memory. The code consists of main routine as well as the interrupt service routine that perform the required job. Note that the main code performs displaying character continuously. While the interrupt service routine will interrupt the main routine and acquire updated data, then storing onto a memory. The source code is attached in Appendix A. Some features have been added to the receiver which is the LED indicator for the ease of debugging. There are two LED's have been added (red and orange). Red LED indicates an error in writing or reading process from a memory or if the memory is not working. While the orange LED indicates the process of downloading data.

The source code is written in **.asm* file. The software used is MPLAB. To begin write the code the following step is applied:

1. Start the MPLAB software
2. From the toolbar click Project>>New. Fill the project's name and save it in respective folder. Then, click OK.
3. From the toolbar, click File>>New. Then click File>>Save. Save the file as *filename.asm* in the respective folder. Then, click OK.
4. From the toolbar, click Project>>Edit Project. Dialog window will appears. Click Add Node. Select *filename.asm*. Then, click OK
5. Now, the code is ready to be written and compiled.

After the code has been written, it needs to be downloaded to the PIC chip. The steps are as follow:

Before downloading the code, make sure the warp13 hardware is connected to the serial port.

1. From the toolbar, click PICprogrammer>>Enable Programmer
2. Dialog windows will appear. Select the respective microcontroller in processor box.
3. Click Program. Another dialog window will appear. The code has been successfully downloaded after the dialog windows disappear.
4. Now, the code has been downloaded.

In storing the data, the receiver must agree with the transmitter. Which is the receiver is expecting the Start Block to be received first followed by data block. Here the data block type is used to storing the data in the memory. In this case, instead of using address to storing the data, it is using block type as indicator to store the data at certain particular location. Thus, the receiver must check the value of data block (represent by binary code) and then store the data block according at its location (e.g Name Block will be stored in name location). Table 3.1 illustrates how data is stored in the memory

Table 3.1: Memory Map

Page A0

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	L	e	c	t	u	r	e	r	:							
10	M	o	h	d		Z	u	k	I							
20	E	x	t	.		N	o	:		7	8	0	7			
30	C	l	a	s	s		S	e	s	s	i	o	n	:		
40	S	l	o	t		1	:	M	i	c	r	o	P			
50	M	o	n		9	:	0	0	-	1	1	:	0	0		
60	2	2	-	0	2	-	1	7								
70	S	l	o	t		2	:	M	i	c	r	o	P			
80	W	e	d		1	0	:	0	0	-	1	2	:	0	0	
90	2	2	-	0	2	-	1	7								
A0	C	o	n	s	t	.		H	o	u	r	:				
B0	M	o	n		1	4	:	0	0	-	1	5	:	0	0	
C0	T	h	u		9	:	0	0	-	1	0	:	0	0		
D0																
E0																
F0																

Table 3.1 continue

Page A2

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	T	e	s	t	:											
10	C	o	u	r	s	e										
20	D	a	y		T	i	m	e								
30	D	a	t	e												
40	V	e	n	u	e											
50	A	w	a	y		M	e	s	s	a	g	e	:			
60																
70																
80																
90																
A0																
B0																
C0																
D0																
E0																
F0																

The memory consists of 4 pages which each page can be occupied with 256 bytes. For this project only two pages is used. The page is designated as A0, A2, A4, A6.

From Table 3.1 it can be visualized how the data is stored. Note that the Page A0 and A2 is the address of the memory. The number at the upper row and the most left column is the subaddress. Thus, to read on the particular location, it needs to specify the page address followed by subaddress. While to write onto the location it needs the page address followed by subaddress and followed by data to be written.

5. **Testing the hardware as well as the code – the circuit connection is correct and the code should perform the sequence**

   In testing the hardware (receiver and transmitter) occur no problems or no short circuit. For the receiver, the LCD connection has no problem. The microcontroller is being able to perform the main routine as well as the interrupt service routine. In debugging the main routine, it is expected to read character form a memory and display the character. If it displays the correct character, thus, the main routine does not have any error. In debugging the interrupt routine, one interrupt signal is used to interrupt the microcontroller. It is expected to exit from main routine and perform the interrupt service routine. In both debug processes, both routines appear no error.

6. **Program the PIC16F84 for manipulate data from parallel port and send (generating) using infra red protocol**

   The microcontroller gets the data from the PC, and then transmits the data using infrared protocol. The source code is attached in Appendix B. In transmitting the data, there are two processes must be undergone. First process is passing a value from a computer to the microcontroller via parallel port. Secondly, the microcontroller transmits the data to the receiver. The idea in

transmitting the data is that, data is transmitted using block. Thus, before transmitting data block, it must send the start block.

In passing the value from a PC to the microcontroller, a simple protocol has been developed. The protocol is illustrated in Figure 3.3.

S	DATA0	P	S	DATA1	P	S	.....	.....	S	DATA*n*	P

FIGURE 3.3: Data Block

The protocol will start with start bit (S), then follow by DATA0 and stop bit (P). Then it repeats until DATA*n*. Then the PC needs to wait for the transmitter to transmit the Data Block before it sends new Data Block. Before the Data Block is sent, the Start Block must be sent first. This Start Block indicates the **type** of Data Block, **number** of Data Block to be sent, and the number of **byte** in the Data Block. The Start Block is illustrated in Figure 3.4.

S	BLK TYPE	P	S	No of BLK	P	S	No of BYTE	P

FIGURE 3.4: Start Block

In generating the infrared protocol, modified SIRCS is used to transmit a character (1 byte). Figure 3.5 illustrates the modified SIRCS protocol called as IDC Infrared Protocol.

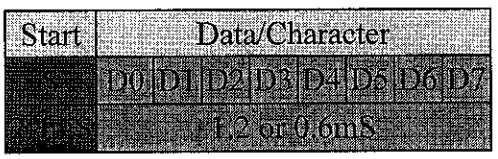


FIGURE 3.5: IDC Infrared Protocol

In transmitting the data, the transmitter follows the same protocol which is it must transmit Start Block followed by Data Block. In this case, the data is

transmitted serially via infrared. Now, the IDC Infrared Protocol is applied. It will transmit Start Pulse followed by one character (8 bits). If there is *n* character to be transmitted, it follows the same protocol which it starts send Start Pulse follows by a character and it is repeating until all *n* character is transmitted. After all blocks have been transmitted, the transmitter will send the Stop Block to indicate the end of the transmission process. The Stop Block consist the Stop Byte. Table 3.2 show the Hex value for each block type.

TABLE 3.2: Hex Code for Each Block

Hex Code	Block Type	Hex Code	Block Type
00h	Lect	0Dh	Day3
01h	Ext	0Eh	Time3
02h	Slot1	0Fh	Const2
03h	Course1	10h	Day4
04h	Day1	11h	Time4
05h	Time1	12h	Test
06h	Venue1	13h	Course5
07h	Slot2	14h	Day5
08h	Course2	15h	Date
09h	Day2	16h	Time5
0Ah	Time2	17h	Venue5
0Bh	Venue2	18h	AwyMsg
0Ch	Const1	AAh	Stop

7. **Testing the transmitter hardware – the circuit connection is correct and be able to transmit the infrared protocol**

   The transmitter does not have any problem in connection and be able to transmit or emit the infrared. The confirmation of the signal transmitted was verified using oscilloscope. The main routine developed for the transmitter should be able to generate the IDC Infrared Protocol signal. The signal observed in debugging process is the Start Pulse, data bit '1' signal and data bit '0' signal. If those signals are generated according to the timing diagram in Figure 2.5, thus the main routine for the transmitter is correct and just need some manipulation in transmitting a sequence on data bit '1' and '0'.

8. **Developing the software – take an input from user, pass to the micro controller via parallel port**

The Visual C++ (Microsoft Foundation Class – MFC) is one of the applications in Visual C++. In this application the Windows for graphical user interface (GUI) has been developed by this program. For this application it uses the dialog based windows. After developing this Windows, the programmer can add any control buttons on this Windows and add any functions code in its source code file.

The variables must be declared as m_*variablename*. m_ indicates that it is a member variable for this class. While the control button (Transmit) is a function performing outputting the data to parallel port. This function need to be defined by the programmer. This interface is developed for testing purpose. The source code is attached in Appendix C.

Steps taken to create a simple dialog based Windows by using Visual C++ (MFC).

a. Run Microsoft Visual C++ 6.0
b. On the click File>>New. The Tab Windows will appear. Click on Project. Then select MFC AppWizard(exe). Fill in the project name under the Project name:. Then, click OK.
c. Select Dialog Based, and then click Finish.
d. The dialog based Windows now has been developed.

The GUI for this project is as shown in Figure 3.6.

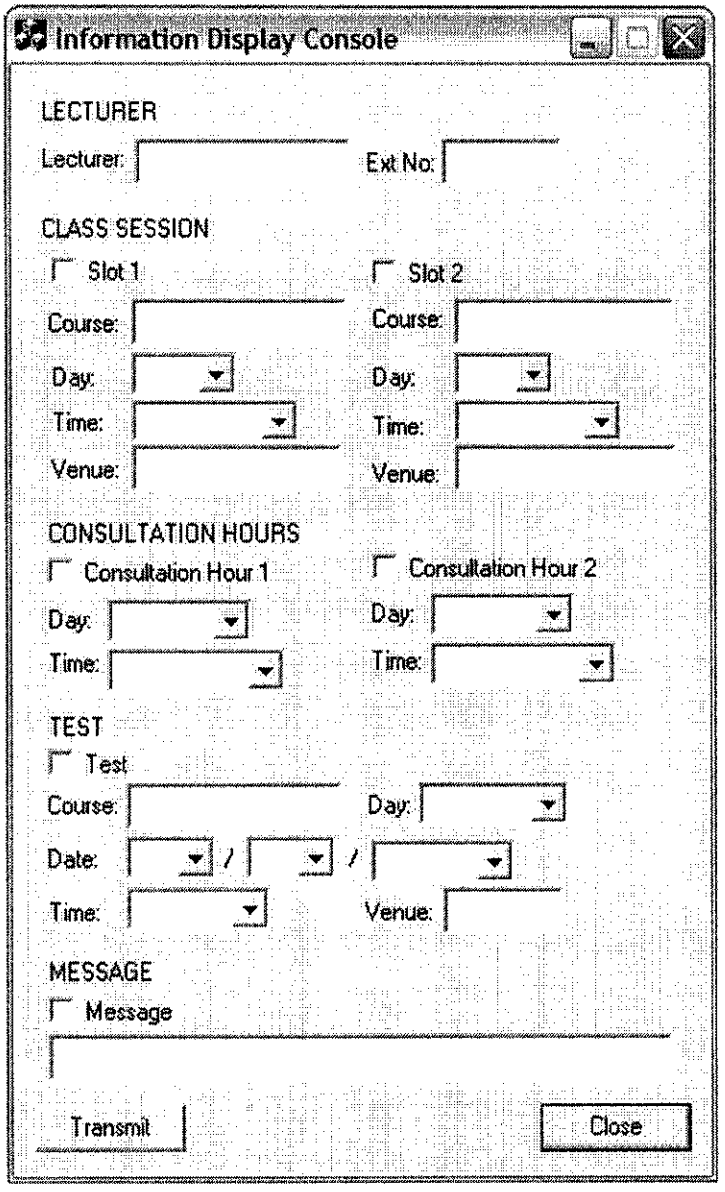


FIGURE 3.6: Graphical User Interface

9. **Integrate testing between transmitter and receiver – transmitting a character, then the receiver should write the character onto memory and be able to read and display the respective character**

   The full system is integrated and supposed to perform the routine properly. The display should be able to display lecturer's name, extension number, class session, consultation hour and etc. continuously. The receiver also should be able to handle the interrupt when there is a need to update the information.

The software should be able to passing data entered by user to the transmitter. The transmitter should be able to transmit the data according to the defined protocol.

3.2 Tool

Tools/equipment required to complete the tasks::

1. PIC16F84 microcontroller & datasheet
2. PIC16F871 microcontroller & datasheet
3. Infra Red LED
4. 16 MHz oscillator
5. 4 MHz oscillator
6. ST24C08 – Serial 8K EEPROM & datasheet
7. LCD Display (5x7 dots) & datasheet
8. 38 kHz infrared filter
9. Transistor (2 X 2N3904 & 2N3906)
10. Oscilloscope
11. Multimeter
12. PICStart/Warp13
13. MPLab Software
14. Visual C++ Software

CHAPTER 4

RESULTS AND DISCUSSION

This section discusses the debugging process of the system. Firstly, it focuses into the hardware followed by the software. In the hardware, it focuses on the connection of the circuit. While in debugging the software, it focuses on the ability of the software to passing the parameter to the transmitter. Furthermore, it discusses the confirmation testing of the system by observing the signal generated by the transmitter.

4.1 The hardware

The construction of the hardware has been done. The hardware does not have any short circuit. The hardware for receiver and transmitter are shown in Figure 4.1 and Figure 4.2 respectively.

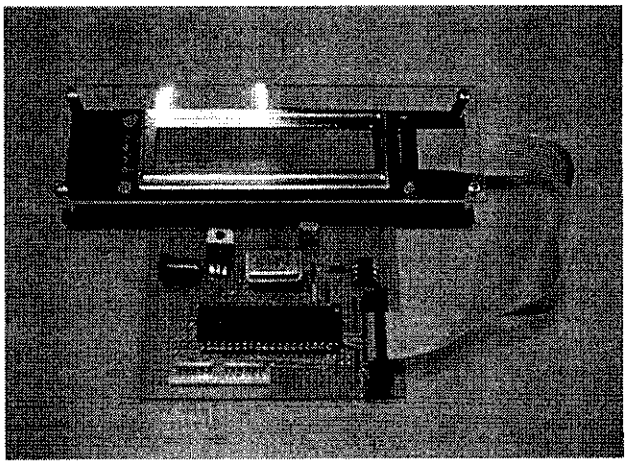


Figure 4.1: Receiver Circuit

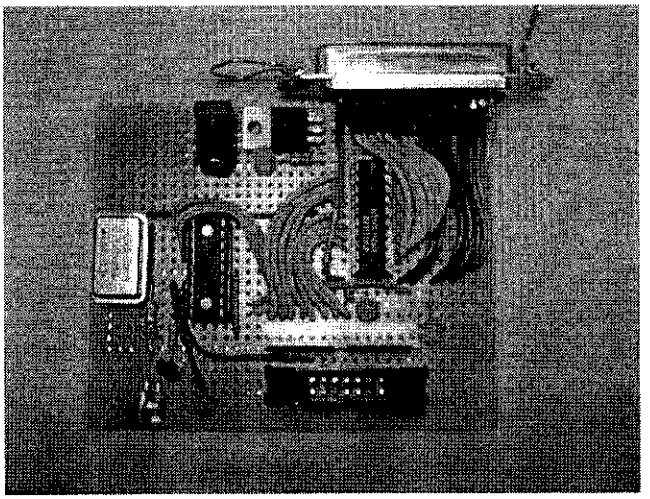


Figure 4.2: Transmitter Circuit

4.2 Debugging

For the receiver there is no problem in connection. However, at the beginning there is a problem regarding the I^2C protocol. Supposed it performs writing a character to the NVM, read the character and display on the LCD. The problem is no character is displayed. The problem is discussed in the discussion section.

For the transmitter, the signal generated was observed using oscilloscope. For the debugging process, the transmitter must be able to generate the signal as shown in Figure 2.4.

Transmitted waveform

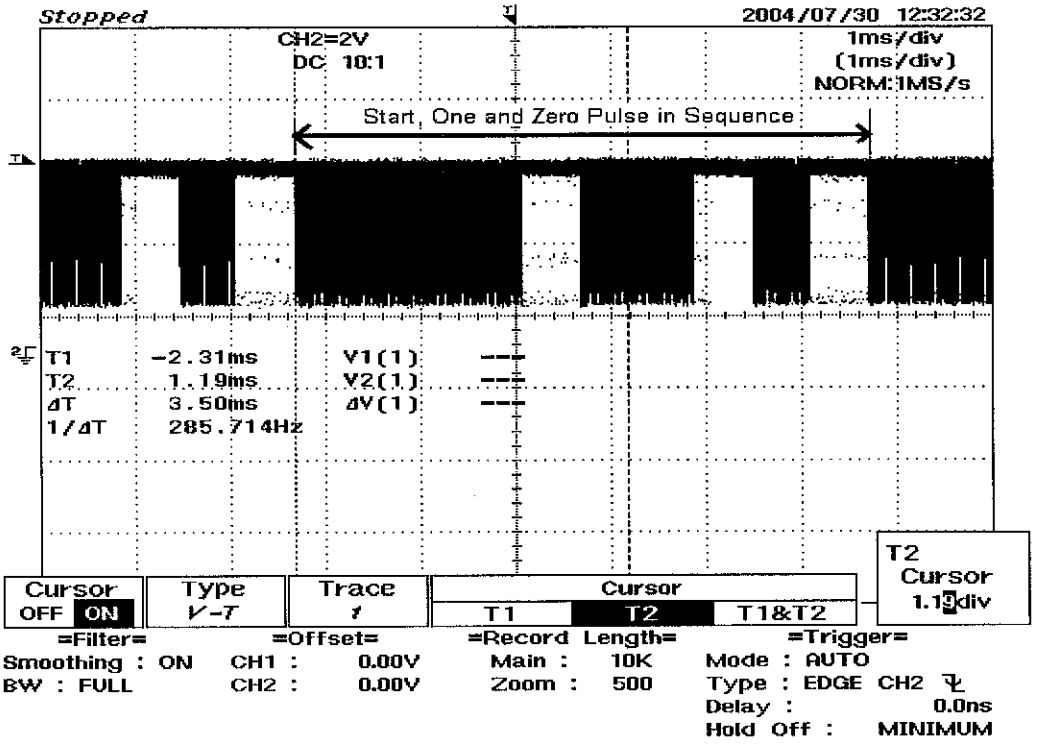


Figure 4.3: Start Pulse, Data Bit '1' Pulse, Data Bit '0' Pulse in Sequence

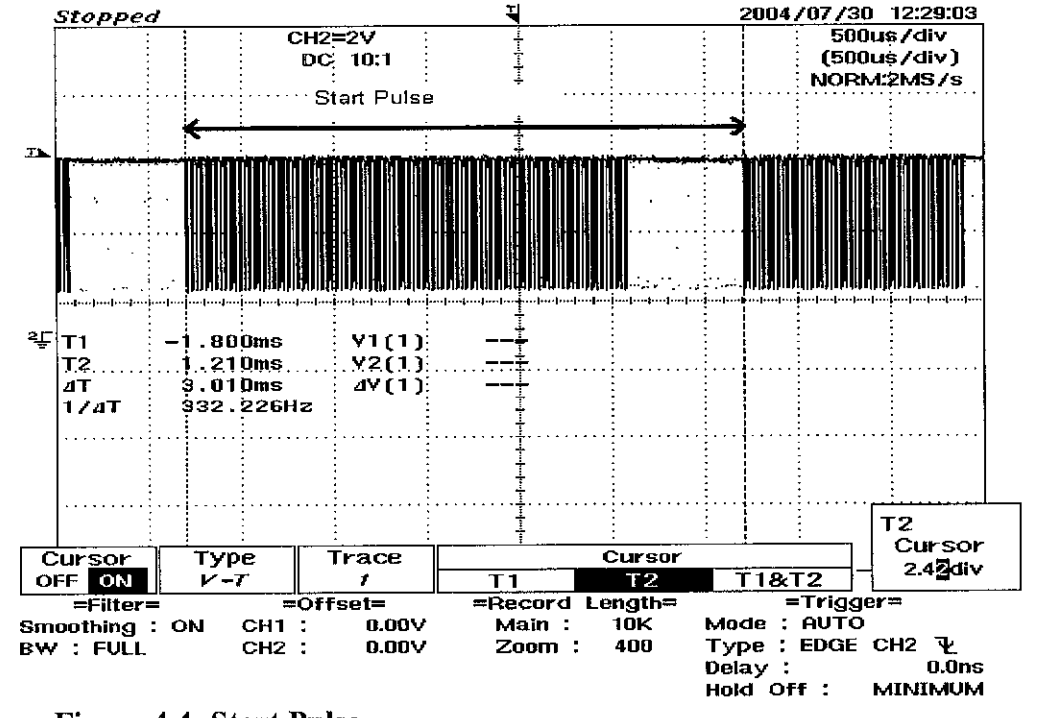


Figure 4.4: Start Pulse

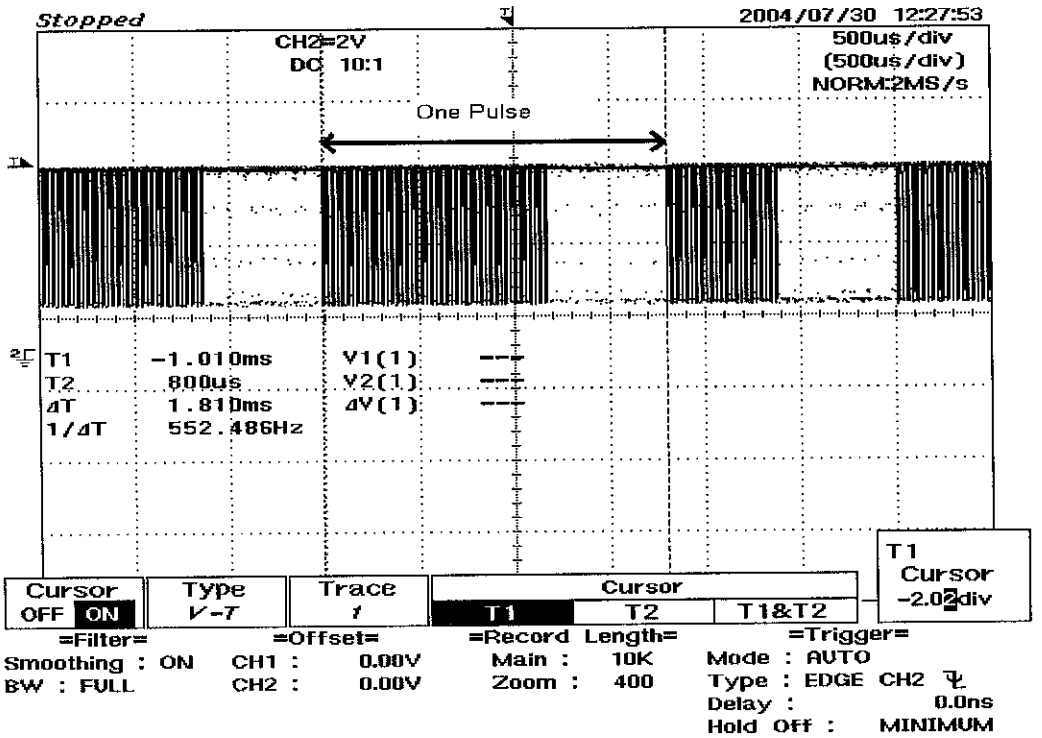


Figure 4.5: Data Bit '1' Pulse

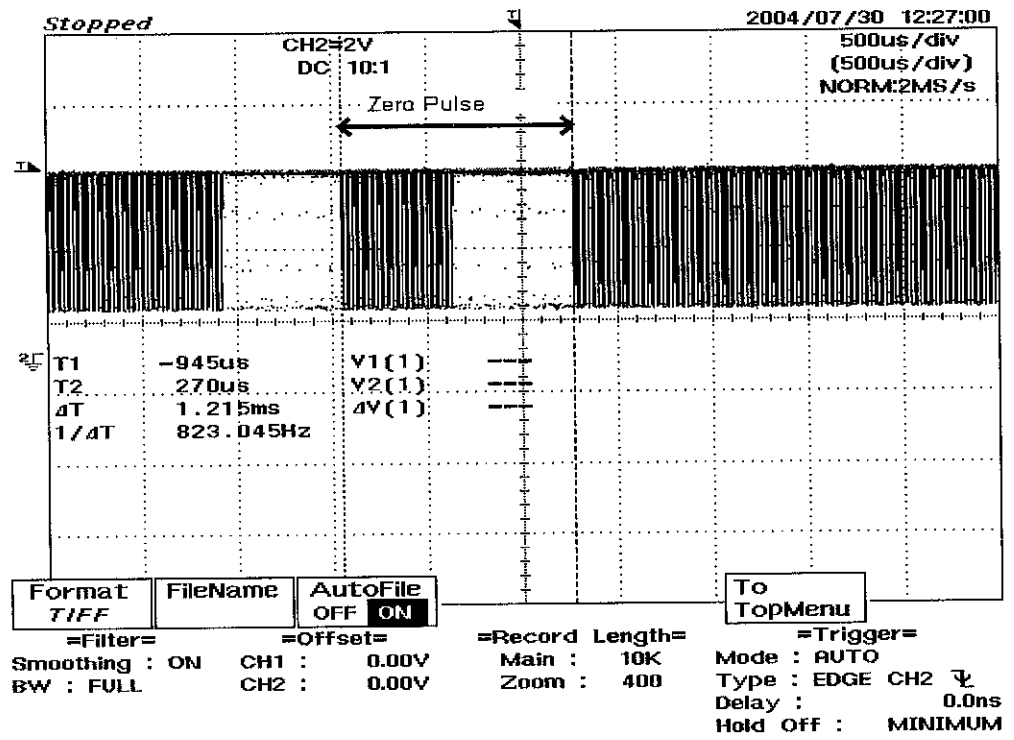


Figure 4.6: '0' Pulse

Comment: From Figure 4.3, it shows that the microcontroller is being able to transmit Start Pulse, data bit '1' Pulse and data bit '0' Pulse. The dark area is actually is the 38 kHz carrier that is used to transmit the signal. From Figure 4.4, it shows the Start Pulse of duration 3.01 mS. This duration is almost as desired Start Pulse where the theory value is 2.4mS for HIGH state and 0.6mS for LOW state which the sum is 3.00 mS. From Figure 4.5 and Figure 4.6, it shows the '1' Pulse and '0' Pulse respectively. The '1' Pulse duration is 1.81mS and '0' Pulse duration is 1.21mS. Both durations also are close to the theory value which are 1.8mS for '1' Pulse and 1.2mS for '0' Pulse. Thus, from this waveform it is confirmed that the microcontroller is capable of generating those three signals and the code is assumed no error.

In debugging the software, the dummy interface was developed. This dummy software is simple software whereby it passes a character to the transmitter, and then the transmitter transmits the character. The purpose of this dummy software is to confirm whether the software is being able to pass the value to the transmitter. Then, the transmitter on the other hand transmits the character using the defined protocol. Figure 4.7 shows the dummy software.

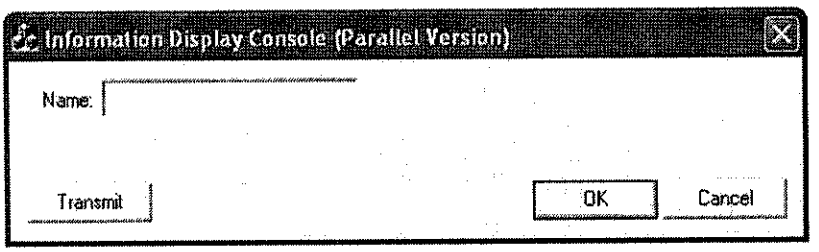


FIGURE 4.7: Dummy Software

In confirmation of the data transmitted by the transmitter, the signal was observed using oscilloscope. There are three characters were transmitted which are character 'A', 'B', and 'C'. The following figures show the signal captured from the oscilloscope.

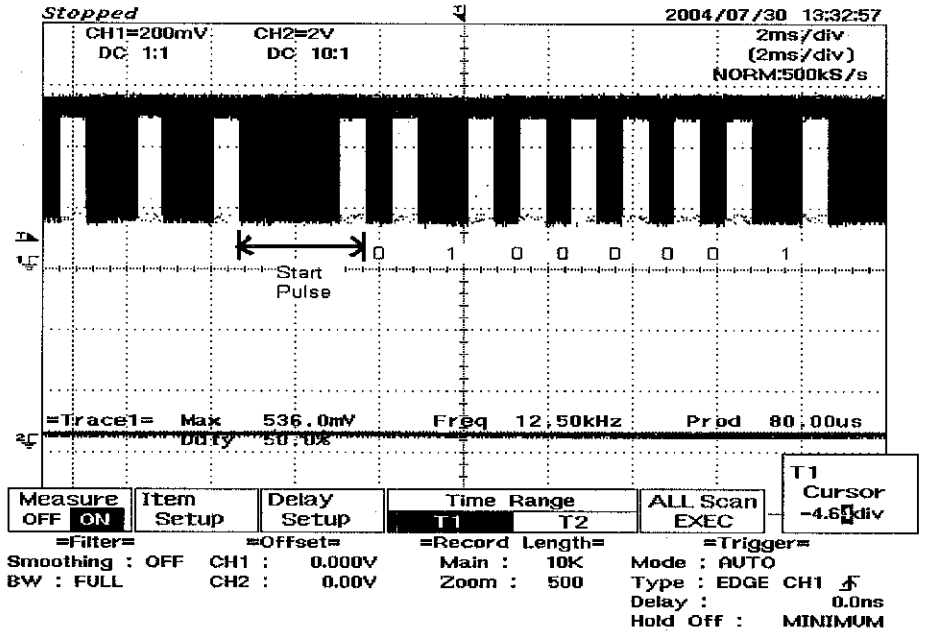


Figure 4.8: Character A (ASCII 0x41)

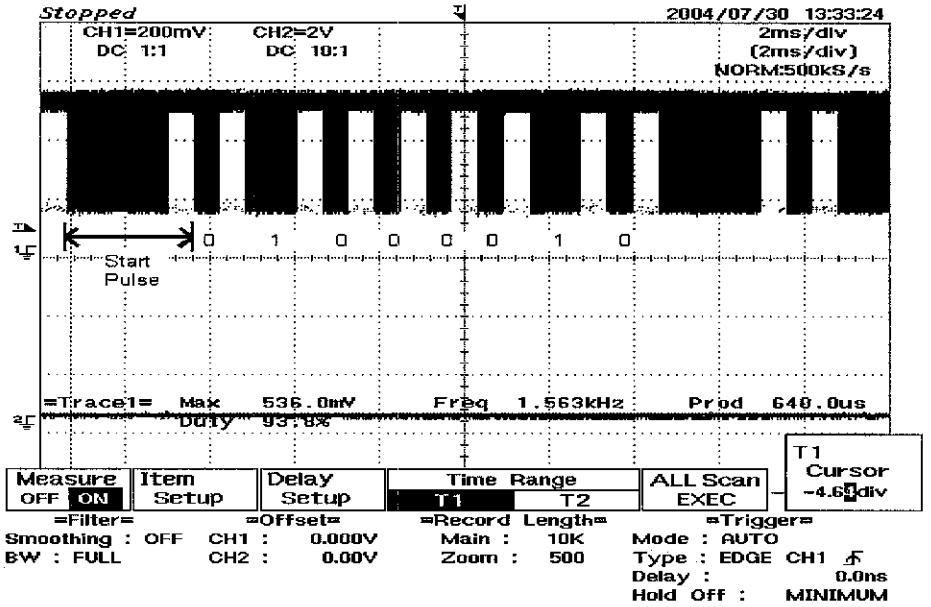


Figure 4.9: Character B (ASCII 0x42)

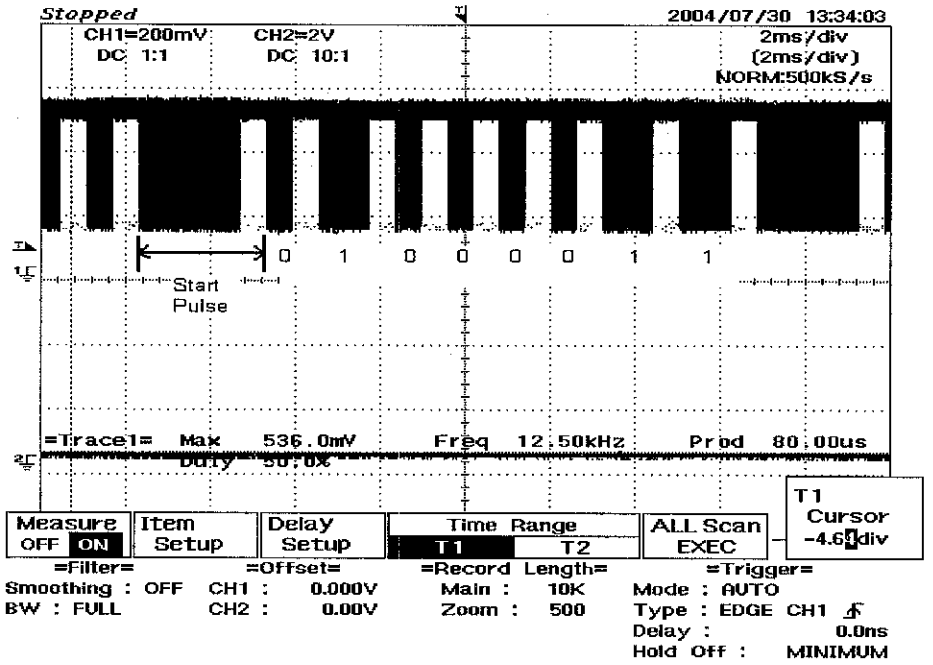


Figure 4.10: Character C (ASCII 0x43)

Comment: From Figure 4.8, Figure 4.9, and Figure 4.10, it shows that the ASCII for all the characters was transmitted. The value is the same as the represent in the ASCII table in Table 4.1. Thus, this confirmed that the coding for transmitting the ASCII for the character is correct and is assumed to be no error.

Table 4.1: ASCII Table

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.asciitable.com

4.3 Discussion

For the receiver, it must be able to sampling, storing and displaying data transmitted. It must display the data continuously. Some features have been added to the receiver. There are LED's to indicate power ON, failure in memory and storing data. As users plug in the power supply, green LED will be ON. This indicates the hardware is working and the hardware should display all the particulars discussed before. When a user transmits updated data from a PC, the receiver's routine is interrupted. While downloading the data it will display a phrase 'Downloading...'. At the same time the orange LED will be ON and OFF as the block of data is transmitted to indicate the block is being storing onto memory. Right after downloading process is finish, it will display a phrase 'Completed'. After that it will continue with its main routine.

For the transmitter, it must be able to generate the infrared signal. It also must be able to take data from a PC and then transmits to the receiver.

For the software it must be able to pass the data to the transmitter. It also must generate the protocol in which the transmitter could understand. It is also play a role in generating a protocol for infrared transmission. The software does not have to be installed. Users just need to copy the software onto their PC. However, there are some files to be copied onto their system. There are *MFC24D.DLL, MFC024D.DLL,* and *MSVCRTD.DLL.* For Windows 98 users, those files should be copied onto *Windows/System* directory. For Windows XP users, those files should be copied onto *Windows/System32* directory. However, if their PC has been equipped with Microsoft Visual C++ software, they may have to skip the procedure.

CHAPTER 5

CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The Information Display Console (IDC) may convey a message to the others such as lecturer's name, extension number, class session, consultation hour, test to be conducted, as well as away message. With the IDC it can improve the communication among lecturers and students. It also helps other, at least know the lecturer's schedule. It also helps to reduce the paper usage and can be a substitute especially in putting on a message.

5.2 Recommendation

The IDC should be installed to enhance the communication among others. However, the IDC still can be improved by adding some modification especially to its appearance. Both the transmitter and receiver circuits can be compacted by using surface mounted components with well designed printed circuit board. This is to reduce the size of the hardware. Furthermore, the hardware itself could be equipped with proper housing to give a nice finishing.

Secondly, further study should be conducted on the receiver about an alternative power supply in case of black-out. In this case, one might consider the battery-powered as the alternative. Thus, it must take into account the power consumption, switching mechanism, and for how long the system can stand in determining and designing the new battery-powered system.

References

1. Optrex Corporation, *Dot Matrix Character LCD Module User's Manual,* <http://www.apollodisplays.com>
2. Cadence Design Foundry (UK) Ltd., *Inter Integrated Circuit (I2C), Technical Data Sheet, Part Number: T-CS-PE-0007-100, Document Number: I-IAP01-0045-USR Rev 09,* <http://www.cadence.com/datasheets/I2C_DataSheet.pdf>
3. SGS-Thomson Microelectronics, *ST24/25C08, ST24C08R, ST24/25W08, SERIAL 8K (1K x 8) EEPROM,* Data Sheet.
4. Microchip Website, *Application Notes: TB 062 – High Power IR LED Driver Using the PIC16C781/782,* <http://www.microchip.com>
5. Nigel's Goodwin, *WinPicProg1.91, PIC Tutorial Five – Infrared Communication,* <http://www.winpicprog.co.uk/pic_tutorial5.htm>
6. Janet Louise Axelson, Jan Axelson, *Parallel Port Complete,* <http://www.lvr.com>
7. *Interfacing to the IBM-PC Parallel Printer Port,* <http://www.doc.ic.ac.uk/~ih,doc/par/>
8. 2002 Lava Computer MFG Inc., *IEEE 1284: Parallel Port,* <http://www.lavalink.com>
9. John B. Peatman, *DESIGN with PIC MICROCONTROLLERS,* Prentice Hall, Upper Saddle River, New Jersey, 1998
10. Microchip Website, *Datasheet PIC16F870/871 – 28/40-Pin 8-Bit CMOS Flash Microcontrollers,* <http://www.microchip.com>
11. Microchip Website, *Datasheet PIC16F8X – 18-Pin Flash/EEPROM 8-Bit Microcontroller,* <http://www.microchip.com>
12. Microchip Website, *PICmicro MID – Range MCU Family Reference Manual,* <http://www.microchip.com>
13. Microchip Website, *Application Notes: AN566 – Using the PortB Interruprt on Change as an External Interrupt,* < http://www.microchip.com>
14. David I. Schneider, *An Introduction to Programming Using Visual Basic 6.0.* Fourth edition, Prentice-Hall, 1995

APPENDIX A

Receiver Code

```
;-------------------------------------------------------------------------------------------------------------------
;
;       Programmer: Ahmad Rizal bin Muhammad Arif
;       ID No: 1922
;       Program: Electrical & Electronics Engineering
;       Company: Universiti Teknologi PETRONAS
;
;       Note: This code is the receiver code. The routine is performing sampling data transmitted, storing and displaying
;             character
;
;-------------------------------------------------------------------------------------------------------------------

#include <p16f871.inc>

        ERRORLEVEL      0,-302      ;suppress bank selection messages
        __config _LVP_OFF & _XT_OSC & _WDT_ON & _PWRTE_ON & _CP_OFF & _BODEN_OFF & _DEBUG_OFF

;-------------------------Reserving register ------------------------

cblock  0x20                ;start of general purpose registers
        LoX                 ;
        Bit_Cntr            ;
        Blk_Type            ;
        Blk_No              ;
        Byte_No             ;
        Data_Byte0          ;
        Data_Byte1          ;
        Data_Byte2          ;
        Data_Byte3          ;
        Flags               ;
        Flags2              ;
        Flags3              ;
        Flags4              ;
        FlagsR              ;
        Flags5              ;
        tmp                 ;
        tmp1                ; temporary storage
        tmp2                ;
        tmp3                ;
        tmp4                ;


        Byte_NoLect         ;
        Byte_NoExt          ;

        Byte_NoSlot1        ;
        Byte_NoCourse1      ;
        Byte_NoDay1         ;
        Byte_NoTime1        ;
        Byte_NoVenue1       ;

        Byte_NoSlot2        ;
        Byte_NoCourse2      ;
        Byte_NoDay2         ;
        Byte_NoTime2        ;
        Byte_NoVenue2       ;

        Byte_NoConst1       ;
        Byte_NoDay3         ;
        Byte_NoTime3        ;

        Byte_NoConst2       ;
```

```
            Byte_NoDay4         ;
            Byte_NoTime4        ;

            Byte_NoTest         ;
            Byte_NoCourse5      ;
            Byte_NoDay5         ;
            Byte_NoDate         ;
            Byte_NoTime5        ;
            Byte_NoVenue5       ;

            Byte_NoAwyMsg       ;

            byte                ;
            i2c_freq            ;
            nbit                ;
            w_temp              ;
            w_temp2             ;
            w_temp3             ;
            waitLCD             ;
            W_TEMP              ;
            STATUS_TEMP         ;
            tmp5
            PCLATH_TEMP
endc
;-------------------------------------------------------------------------

;-------------------------------------------------------------------------
IR_PORT             Equ     PORTA
TRISD               Equ     0x88
;-------------------------------------------------------------------------

;---------------------- Defining Value for variable ----------------------
#define     IR_In                       3
#define     RS                          7
#define     ENABLE                      6
#define     RW                          5
#define     LCD_CLR_DISPLAY             0x01    ;0000 0001
#define     LCD_CURSOR_HOME             0x02    ;0000 0010
#define     LCD_ENTRY_MODE_SET          0x06    ;0000 0110
#define     LCD_CURSOR                  0x10    ;0001 0000
#define     LCD_FUNCTION_SET            0x38    ;0011 1000
#define     LCD_INIT_CGRAM              0x40    ;0100 0000
#define     LCD_INIT_DDRAM              0x80    ;1000 0000
#define     LCD_INIT                    0x0C    ;0000 1101
#define     WAITLCD                     0x05    ;0000 0101

#define     ErrFlag                     0       ;Flags used for received bit
#define     StartFlag                   1
#define     One                         2
#define     Zero                        3
#define     StopFlag                    4
#define     Update                      0

#define     Lect_Blk                    0x00
#define     Ext_Blk                     0x01

#define     Slot1_Blk                   0x02
#define     Course1_Blk                 0x03
#define     Day1_Blk                    0x04
#define     Time1_Blk                   0x05
#define     Venue1_Blk                  0x06

#define     Slot2_Blk                   0x07
#define     Course2_Blk                 0x08
#define     Day2_Blk                    0x09
#define     Time2_Blk                   0x0A
#define     Venue2_Blk                  0x0B

#define     Const1_Blk                  0x0C
#define     Day3_Blk                    0x0D
```

```
#define    Time3_Blk                0x0E

#define    Const2_Blk               0x0F
#define    Day4_Blk                 0x10
#define    Time4_Blk                0x11

#define    Test_Blk                 0x12
#define    Course5_Blk              0x13
#define    Day5_Blk                 0x14
#define    Date_Blk                 0x15
#define    Time5_Blk                0x16
#define    Venue5_Blk               0x17

#define    AwyMsg_Blk               0x18
#define    Stop_Blk                 0xAA

#define    LectFlag                 0        ;Flags used to determine type of block
#define    ExtFlag                  1

#define    Slot1Flag                2
#define    Course1Flag              3
#define    Day1Flag                 4
#define    Time1Flag                5
#define    Venue1Flag               6

#define    Slot2Flag                7
#define    Course2Flag              0
#define    Day2Flag                 1
#define    Time2Flag                2
#define    Venue2Flag               3

#define    Const1Flag               0
#define    Day3Flag                 1
#define    Time3Flag                2

#define    Const2Flag               3
#define    Day4Flag                 4
#define    Time4Flag                5

#define    TestFlag                 0
#define    Course5Flag              1
#define    Day5Flag                 2
#define    DateFlag                 3
#define    Time5Flag                4
#define    Venue5Flag               5

#define    AwyMsgFlag               6
#define    Stop_Flag                7

#define    WREG                     0
#define    I2C_PORT                 PORTA
#define    SDA_IN                   2
#define    HIGH_HIGH                0x03          ; 0000 0011
#define    HIGH_LOW                 0x02          ; 0000 0010
#define    LOW_LOW                  0x00          ; 0000 0000
#define    LOW_HIGH                 0x01          ; 0000 0001

;-------------------------------------------------------------------------------

           org      0x00
           goto     Setup

           org      0x04
           goto     ISR
```

```
;----------------------------Interruprt Service Routine---------------------------------

ISR
Push    nop
        nop
        MOVWF   W_TEMP          ; Copy W to TEMP register
        SWAPF   STATUS,W        ; Swap status to be saved into W
        CLRF    STATUS          ; bank 0, regardless of current bank, Clears IRP,RP1,RP0
        MOVWF   STATUS_TEMP     ; Save status to bank zero STATUS_TEMP register
        MOVF    PCLATH, W       ; Only required if using pages 1, 2 and/or 3
        MOVWF   PCLATH_TEMP     ; Save PCLATH into W
        CLRF    PCLATH          ; Page zero, regardless of current page
        Clrf    FlagsR
        call    delay4
        call    ISR_Code


;----------------------------Interrupt Service Routine End-----------------------------------

;------------------------ Setup Microcontroller ----------------------------
        org     0xD0            ; Start of Setup routine
Setup   BCF     STATUS, RP0     ;
        BCF     STATUS, RP1     ; Bank0
        CLRF    PORTA           ; Initialize PORTA by clearing output data latches
        BSF     STATUS, RP0     ; Select Bank 1
        MOVLW   0x06            ; Configure all pins
        MOVWF   ADCON1          ; as digital inputs
        MOVLW   0x0C            ; Value used to initialize data direction
        MOVWF   TRISA           ; Set RA<3:0> as inputs RA<5:4> as outputs TRISA<7:6> are always
                                ; read as '0'.
        movlw   0x90            ; Value to enable Global and External Interrupt
        movwf   INTCON          ; Enable Global Interrupt and External Interrupt
        bcf     OPTION_REG, 6   ; Enable Edge Trigger for External Interrupt
        bcf     INTCON, 1

        movlw   0x01            ; Value used to initialize data direction
        movwf   TRISB           ; Set Port B<7:1> as an output and PortB<0> as an input
        clrf    TRISC           ; Set Port C as an output
        clrf    TRISD           ; Set Port D as an output

        bcf     STATUS, RP0     ; Select Bank 0
        call    InitLCD         ; LCD Initialization
        clrf    PORTA           ; Clearing Content of PORTA
        clrf    PORTB           ; Clearing Content of PORTB
        bsf     PORTC, 0        ;
        bsf     PORTC, 3        ;
        clrf    PORTD           ; Clearing Content of PORTD
        clrf    Data_Byte0      ; Clearing Content of Data_Byte0 Register
        clrf    Data_Byte1      ; Clearing Content of Data_Byte1 Register
        clrf    Data_Byte2      ; Clearing Content of Data_Byte2 Register
        clrf    Data_Byte3      ; Clearing Content of Data_Byte3 Register
        clrf    tmp1            ; Clearing tmp1 Register
        clrf    Flags           ; Clearing Flags Register
        clrf    Flags2          ; Clearing Flags2 Register
        clrf    Flags3          ; Clearing Flags3 Register
        clrf    Flags4          ; Clearing Flags4 Register
        clrf    Flags5          ; Clearing Flags5 Register
        goto    main
;---------------------------------- End ----------------------------------------
```

```
;---------------------- Main Routine ----------------------------------
        org     0x100                   ;
main    clrwdt                          ; Clearing watchdog timer to avoid internal reset
        btfsc   FlagsR, LectFlag        ; Check Lecturer Name Flag. Display if Set
        call    RLect                   ; Display Lecturer Name
        btfsc   FlagsR, ExtFlag         ; Check Ext. No Flag. Display if Set
        call    RExt                    ; Display Ext. No.
        btfsc   FlagsR, Slot1Flag       ; Check Slot 1 Flag. Display if Set
        call    RSlot1                  ; Display Slot 1 (Class Session)
        btfsc   FlagsR, Slot2Flag       ; Check Slot 2 Flag. Display if Set
        call    RSlot2                  ; Display Slot 2 (Class Session)
        btfsc   FlagsR, 4               ; Check Consultation Hour 1 Flag. Display if Set
        call    RConst1                 ; Display Consultation Hour 1
        btfsc   FlagsR, Const2Flag      ; Check Consultation Hour 2 Flag. Display if Set
        call    RConst2                 ; Display Consultation Hour 2
        btfsc   FlagsR, 5               ; Check Test Flag. Display if Set
        call    RTest                   ; Display Test.
        btfsc   FlagsR, AwyMsgFlag      ; Check Away Message Flag. Display if Set
        call    RAwyMsg                 ; Display Away Message
        goto    main                    ; Go to Main
;---------------------------------------------- Main Routine End ----------------------------------------------

;-----------------------------------------------Interrupt Service Routine Function----------------------------------------
ISR_Code
        movlw   0x01    ; Value used to Clear Display
        call    InstLCD ; Clearing Display
        movlw   0x83    ; Value used determine location at display
        call    InstLCD ; Location at display
        movlw   'D'     ;
        call    DataLCD ;
        movlw   'o'     ;
        call    DataLCD ;       Displaying
        movlw   'w'     ;
        call    DataLCD ;       Downloading....
        movlw   'n'     ; ---------------------------------------
        call    DataLCD ;
        movlw   'l'     ;       This is to indicate the beginning of sampling and storing processes
        call    DataLCD ;
        movlw   'o'     ;
        call    DataLCD ;
        movlw   'a'     ;
        call    DataLCD ;
        movlw   'd'     ;
        call    DataLCD ;
        movlw   'i'     ;
        call    DataLCD ;
        movlw   'n'     ;
        call    DataLCD ;
        movlw   'g'     ;
        call    DataLCD ;
        movlw   '.'     ;
        call    DataLCD ;
        movlw   '.'     ;
        call    DataLCD ;
        movlw   '.'     ;
        call    DataLCD ;
        movlw   '.'     ;
        call    DataLCD ;
        movlw   0xC0    ;
        call    InstLCD ;
dash    movlw   0x14    ;
        movwf   tmp     ;
_dash   clrwdt          ;
        movlw   '-'     ;
        call    DataLCD ;
        decfsz  tmp, f  ;
        goto    _dash   ;
```

```
        goto    update              ; (Updating Information )
update  call    GtStBlk             ; Get Start Block
        btfsc   Flags2, LectFlag    ; Check if Lecturer Block is Sent. Write/Store if Flag is Set
        goto    WLect               ; Write/Store the Data
        btfsc   Flags2, ExtFlag     ; Check if Ext No Block is Sent. Write/Store if Flag is Set
        goto    WExt                ; Write/Store the Data
        btfsc   Flags2, Slot1Flag   ; Check if Slot 1 Block is Sent. Write/Store if Flag is Set
        goto    WSlot1              ; Write/Store the Data
        btfsc   Flags2, Course1Flag ; Check if the Course 1 Block is Sent. Write/Store if Flag is Set
        goto    WCourse1            ; Write/Store the Data
        btfsc   Flags2, Day1Flag    ; Check if the Day 1 Block is Sent. Write/Store if Flag is Set
        goto    WDay1               ; Write/Store the Data
        btfsc   Flags2, Time1Flag   ; Check if the Time 1 Block is Sent. Write/Store if Flag is Set
        goto    WTime1              ; Write/Store the Data
        btfsc   Flags2, Venue1Flag  ; Check if the Venue 1 Block is Sent. Write/Store if Flag is Set
        goto    WVenue1             ; Write/Store the Data
        btfsc   Flags2, Slot2Flag   ; Check if the Slot 2 Block is Sent. Write/Store if Flag is Set
        goto    WSlot2              ; Write/Store the Data
        btfsc   Flags3, Course2Flag ; Check if the Course 2 Block is Sent. Write/Store if Flag is Set
        goto    WCourse2            ; Write/Store the Data
        btfsc   Flags3, Day2Flag    ; Check if the Day2 Block is Sent. Write/Store if Flag is Set
        goto    WDay2               ; Write/Store the Data
        btfsc   Flags3, Time2Flag   ; Check if the Time 2 Block is Sent. Write/Store if Flag is Set
        goto    WTime2              ; Write/Store the Data
        btfsc   Flags3, Venue2Flag  ; Check if the Venue 2 Block is Sent. Write/Store if Flag is Set
        goto    WVenue2             ; Write/Store the Data
        btfsc   Flags4, Const1Flag  ; Check if the Consultation Hour 1 Block is Sent. Write/Store if Flag is Set
        goto    WConst1             ; Write/Store the Data
        btfsc   Flags4, Day3Flag    ; Check if the Day 3 Block is Sent. Write/Store if Flag is Set
        goto    WDay3               ; Write/Store the Data
        btfsc   Flags4, Time3Flag   ; Check if the Time 3 Block is Sent. Write/Store if Flag is Set
        goto    WTime3              ; Write/Store the Data
        btfsc   Flags4, Const2Flag  ; Check if the Consultation Hour 2 Block is Sent. Write/Store if Flag is Set
        goto    WConst2             ; Write/Store the Data
        btfsc   Flags4, Day4Flag    ; Check if the Day 4 is Sent. Write/Store if Flag is Set
        goto    WDay4               ; Write/Store the Data
        btfsc   Flags4, Time4Flag   ; Check if the Time 4 is Sent. Write/Store if Flag is Set
        goto    WTime4              ; Write/Store the Data
        btfsc   Flags5, TestFlag    ; Check if the Test Block is Sent. Write/Store if Flag is Set
        goto    WTest               ; Write/Store the Data
        btfsc   Flags5, Course5Flag ; Check if the Course 5 Block is Sent. Write/Store if Flag is Set
        goto    WCourse5            ; Write/Store the Data
        btfsc   Flags5, Day5Flag    ; Check if the Day 5 Block is Sent. Write/Store if Flag is Set
        goto    WDay5               ; Write/Store the Data
        btfsc   Flags5, DateFlag    ; Check if the Date Blcok is Sent. Write/Store if Flag is Set
        goto    WDate               ; Write/Store the Data
        btfsc   Flags5, Time5Flag   ; Check if the Time 5 Block is Sent. Write/Store if Flag is Set
        goto    WTime5              ; Write/Store the Data
        btfsc   Flags5, Venue5Flag  ; Check if the Venue 5 Block is Sent. Write/Store if Flag is Set
        goto    WVenue5             ; Write/Store the Data
        btfsc   Flags5, AwyMsgFlag  ; Check if the Away Message Block is Sent. Write/Store if Flag is Set
        goto    WAwyMsg             ; Write/Store the Data
        btfss   Flags4, Stop_Flag   ; Check if the Stop Block is Sent. Stop Sampling Data if Flag is Set.

        goto    update              ;
        goto    _ISR                ;

_ISR    movlw   0x01                ;
        call    InstLCD             ;
        movlw   0x80                ;
        call    InstLCD             ;
        movlw   'D'                 ;
        call    DataLCD             ;       Displaying a phrase
        movlw   'o'                 ;
        call    DataLCD             ;       -------- Completed --------
        movlw   'n'                 ;
        call    DataLCD             ;       This is to indicate that sampling and storing data has been completed
        movlw   'e'                 ;
        call    DataLCD             ;
        movlw   ' '                 ;
```

```
        call    DataLCD         ;
        movlw   'C'             ;
        call    DataLCD         ;
        movlw   'o'             ;
        call    DataLCD         ;
        movlw   'm'             ;
        call    DataLCD         ;
        movlw   'p'             ;
        call    DataLCD         ;
        movlw   'l'             ;
        call    DataLCD         ;
        movlw   'e'             ;
        call    DataLCD         ;
        movlw   't'             ;
        call    DataLCD         ;
        movlw   'e'             ;
        call    DataLCD         ;
        movlw   'd'             ;
        call    DataLCD         ;
        movlw   ' '             ;
        call    DataLCD         ;
        movlw   '-'             ;
        call    DataLCD         ;
        movlw   '-'             ;
        call    DataLCD         ;
        movlw   '-'             ;
        call    DataLCD         ;
        movlw   '-'             ;
        call    DataLCD         ;
        movlw   '-'             ;
        call    DataLCD         ;
        call    delay3          ;
;-------------------------------------------------------------------

        nop
        nop
        MOVF    PCLATH_TEMP, W  ; Restore PCLATH
        MOVWF   PCLATH          ; Move W into PCLATH
        SWAPF   STATUS_TEMP,W   ; Swap STATUS_TEMP register into W
                                ; (sets bank to original state)
        MOVWF   STATUS          ; Move W into STATUS register
        SWAPF   W_TEMP,F        ; Swap W_TEMP
        SWAPF   W_TEMP,W        ; Swap W_TEMP into W

        bcf     INTCON, 1       ; Enabling Next Interrupt
        nop                     ;
        retfie                  ; Return from Interrupt Routine
;-------------------------------------------------------------------
```

```
;------------------- Read and Displaying all Particulars ----------------------------
RLect   movlw   0x01
        call    InstLCD
        movlw   0x80            ; Display at 1st line
        call    InstLCD         ;
        call    ReadL           ; Read String 'Lecturer: '
        movlw   0x8A            ;
        call    InstLCD         ;
        movlw   0x10            ;
        movwf   w_temp3         ;
        movf    Byte_NoLect, w
        movwf   Byte_No
        call    ReadG           ;
        call    delay2          ;
        return

RExt    movlw   0xC0            ; 2nd Line
        call    InstLCD         ;
        call    ReadE           ; Display String 'Ext No: '
        call    delay3          ;
        return

RSlot1  movlw   0x01            ;
        call    InstLCD         ;
        call    ReadC           ;
        movlw   0xC0            ;
        call    InstLCD         ;
        movlw   0x40            ;
        movwf   w_temp3         ;
        movf    Byte_NoSlot1, w ;
        movwf   Byte_No         ;
        call    ReadG           ;
        call    delay2          ;
        call    RCourse1        ;
        call    delay3          ;
        call    RDay1           ;
        call    delay3          ;
        call    RTime1          ;
        call    delay3          ;
        call    RVenue1         ;
        call    delay3          ;
        return                  ;

RCourse1 movlw  0xC8            ;
        call    InstLCD         ;
        movlw   0x47            ;
        movwf   w_temp3         ;
        movf    Byte_NoCourse1, w ;
        movwf   Byte_No         ;
        call    ReadG           ;
        call    delay2          ;
        return                  ;


RDay1   movlw   0xC8            ;
        call    InstLCD         ;
        call    ClearLCD        ;
        movlw   0xC8            ;
        call    InstLCD         ;
        movlw   0x50            ;
        movwf   w_temp3         ;
        movf    Byte_NoDay1, w  ;
        movwf   Byte_No         ;
        call    ReadG           ;
        call    delay2          ;
        return                  ;

RTime1  movlw   0xC8            ;
        call    InstLCD         ;
        call    ClearLCD        ;
```

```
            movlw   0xC8                ;
            call    InstLCD             ;
            movlw   0x54                ;
            movwf   w_temp3             ;
            movf    Byte_NoTime1, w     ;
            movwf   Byte_No             ;
            call    ReadG               ;
            call    delay2              ;
            return                      ;

RVenue1     movlw   0xC8                ;
            call    InstLCD             ;
            call    ClearLCD            ;
            movlw   0xC8                ;
            call    InstLCD             ;
            movlw   0x60                ;
            movwf   w_temp3             ;
            movf    Byte_NoVenue1, w    ;
            movwf   Byte_No             ;
            call    ReadG               ;
            call    delay2              ;
            return                      ;

RSlot2      movlw   0x01                ;
            call    InstLCD             ;
            call    ReadC               ;
            movlw   0xC0                ;
            call    InstLCD             ;
            call    ClearLCD            ;
            movlw   0xC0                ;
            call    InstLCD             ;
            movlw   0x70                ;
            movwf   w_temp3             ;
            movf    Byte_NoSlot2, w     ;
            movwf   Byte_No             ;
            call    ReadG               ;
            call    delay2              ;
            call    RCourse2            ;
            call    delay3              ;
            call    RDay2               ;
            call    delay3              ;
            call    RTime2              ;
            call    delay3              ;
            call    RVenue2             ;
            call    delay3              ;
            return                      ;

RCourse2    movlw   0xC8                ;
            call    InstLCD             ;
            movlw   0x77                ;
            movwf   w_temp3             ;
            movf    Byte_NoCourse2, w   ;
            movwf   Byte_No             ;
            call    ReadG               ;
            call    delay2              ;
            return                      ;

RDay2       movlw   0xC8                ;
            call    InstLCD             ;
            call    ClearLCD            ;
            movlw   0xC8                ;
            call    InstLCD             ;
            movlw   0x80                ;
            movwf   w_temp3             ;
            movf    Byte_NoDay2, w      ;
            movwf   Byte_No             ;
            call    ReadG               ;
            call    delay2              ;
            return                      ;
```

```
RTime2   movlw   0xC8
         call    InstLCD
         call    ClearLCD
         movlw   0xC8
         call    InstLCD
         movlw   0x84
         movwf   w_temp3
         movf    Byte_NoTime2, w
         movwf   Byte_No
         call    ReadG
         call    delay2
         return

RVenue2  movlw   0xC8
         call    InstLCD
         call    ClearLCD
         movlw   0xC8
         call    InstLCD
         movlw   0x90
         movwf   w_temp3
         movf    Byte_NoVenue2, w
         movwf   Byte_No
         call    ReadG
         call    delay2
         return

RConst1  movlw   0x01
         call    InstLCD
         movlw   0x80
         call    InstLCD
         movlw   0xA0
         movwf   w_temp3
         movf    Byte_NoConst1, w
         movwf   Byte_No
         call    ReadG
         call    delay2
         call    RDay3
         call    RTime3
         call    delay3
         return

RDay3    movlw   0xC0
         call    InstLCD
         movlw   0xB0
         movwf   w_temp3
         movf    Byte_NoDay3, w
         movwf   Byte_No
         call    ReadG
         call    delay2
         return

RTime3   movlw   0xC4
         call    InstLCD
         movlw   0xB4
         movwf   w_temp3
         movf    Byte_NoTime3, w
         movwf   Byte_No
         call    ReadG
         call    delay2
         return

RConst2  movlw   0x01
         call    InstLCD
         movlw   0x80
         call    InstLCD
         movlw   0xA0
         movwf   w_temp3
         movf    Byte_NoConst2, w
         movwf   Byte_No
         call    ReadG
```

```
        call    delay2
        call    RDay4
        call    RTime4
        call    delay3
        return

RDay4   movlw   0xC0
        call    InstLCD
        movlw   0xC0
        movwf   w_temp3
        movf    Byte_NoDay4, w
        movwf   Byte_No
        call    ReadG
        call    delay2
        return

RTime4  movlw   0xC4
        call    InstLCD
        movlw   0xC4
        movwf   w_temp3
        movf    Byte_NoTime4, w
        movwf   Byte_No
        call    ReadG
        call    delay2
        return

RTest   movlw   0x01
        call    InstLCD
        movlw   0x00
        movwf   w_temp3
        movf    Byte_NoTest, w
        movwf   Byte_No
        call    Readg
        call    delay2
        call    RCourse5
        call    delay3
        call    RDay5
        call    RDate
        call    delay3
        call    RTime5
        call    delay3
        call    RVenue5
        call    delay3
        return

RCourse5 movlw  0xC0
        call    InstLCD
        movlw   0x05
        movwf   w_temp3
        movf    Byte_NoCourse5, w
        movwf   Byte_No
        call    Readg
        call    delay2
        return

RDay5   movlw   0xC0
        call    InstLCD
        call    ClearLCD
        movlw   0xC0
        call    InstLCD
        movlw   0x10
        movwf   w_temp3
        movf    Byte_NoCourse5, w
        movwf   Byte_No
        call    Readg
        call    delay2
        return

RDate   movlw   0xC4
        call    InstLCD
```

```
            movlw   0x14
            movwf   w_temp3
            movf    Byte_NoDate, w
            movwf   Byte_No
            call    Readg
            call    delay2
            return

RTime5      movlw   0xC0
            call    InstLCD
            call    ClearLCD
            movlw   0xC0
            call    InstLCD
            movlw   0x20
            movwf   w_temp3
            movf    Byte_NoTime5, w
            movwf   Byte_No
            call    Readg
            call    delay2
            return

RVenue5     movlw   0xC0
            call    InstLCD
            call    ClearLCD
            movlw   0xC0
            call    InstLCD
            movlw   0x30
            movwf   w_temp3
            movf    Byte_NoVenue5, w
            movwf   Byte_No
            call    Readg
            call    delay2
            return

RAwyMsg movlw   0x01
            call    InstLCD
            movlw   0x07
            call    InstLCD
            movlw   0x94
            call    InstLCD
            movlw   0x40
            movwf   w_temp3
            movf    Byte_NoAwyMsg, w
            movwf   Byte_No
            call    Readg
            call    delay2
            return
;------------------------- End -------------------------------------

;------------ Get Start Block Signal and Check the Parameter ------------

GtStBlk     call    ReadIR          ; Check for the Infra red signal
            call    Get_Data0       ;
            call    delay2          ;

            call    ReadIR          ;
            call    Get_Data0       ;
            call    delay2          ;

            call    ReadIR          ;
            call    Get_Data1       ;
            call    delay2          ;

            call    ReadIR          ;
            call    Get_Data2       ;
            call    delay2          ;

            movf    Data_Byte0, w   ; Storing all Data in Start Block
            movwf   Blk_Type        ; (Type of Block, # of Block, # of Byte)
            movf    Data_Byte1, w   ;
```

```
        movwf   Blk_No          ;
        movf    Data_Byte2, w   ;
        movwf   Byte_No         ;
        goto    Chk_Blk         ; Check Block that has been Transmitted

Chk_Blk clrf    Flags2          ; Clearing Flags
        clrf    Flags3          ;
        clrf    Flags4          ;
        clrf    Flags5          ;

Try_LectBlk     movf    Blk_Type, w     ; the algorithm for checking the Block Type is as follow
                sublw   Lect_Blk        ;
                btfss   STATUS, Z       ; From the start block, the block type byte is stored to a temporary register.
                goto    Try_ExtBlk      ; Then this value is subtract to the assigned value of type of block.
                movf    Byte_No, w      ; If the remainder is zero. Thus, the respective block s correct.
                movwf   Byte_NoLect     ; Thus, certain flag is set according to the type of block
                bsf     Flags2, LectFlag ;
                bsf     FlagsR, LectFlag ;
                retlw   0x00            ;

Try_ExtBlk      movf    Blk_Type, w     ;
                sublw   Ext_Blk         ;
                btfss   STATUS, Z       ;
                goto    Try_Slot1Blk    ;
                movf    Byte_No, w      ;
                movwf   Byte_NoExt      ;
                bsf     Flags2, ExtFlag ;
                bsf     FlagsR, ExtFlag ;
                retlw   0x00

Try_Slot1Blk    movf    Blk_Type, w     ;
                sublw   Slot1_Blk       ;
                btfss   STATUS, Z       ;
                goto    Try_Course1Blk  ;
                movf    Byte_No, w      ;
                movwf   Byte_NoSlot1    ;
                bsf     Flags2, Slot1Flag ;
                bsf     FlagsR, Slot1Flag ;
                retlw   0x00            ;

Try_Course1Blk  movf    Blk_Type, w     ;
                sublw   Course1_Blk     ;
                btfss   STATUS, Z       ;
                goto    Try_Day1Blk     ;
                movf    Byte_No, w      ;
                movwf   Byte_NoCourse1  ;
                bsf     Flags2, Course1Flag ;
                retlw   0x00            ;

Try_Day1Blk     movf    Blk_Type, w     ;
                sublw   Day1_Blk        ;
                btfss   STATUS, Z       ;
                goto    Try_Time1Blk    ;
                movf    Byte_No, w      ;
                movwf   Byte_NoDay1     ;
                bsf     Flags2, Day1Flag ;
                retlw   0x00            ;

Try_Time1Blk    movf    Blk_Type, w     ;
                sublw   Time1_Blk       ;
                btfss   STATUS, Z       ;
                goto    Try_Venue1Blk   ;
                movf    Byte_No, w      ;
                movwf   Byte_NoTime1    ;
                bsf     Flags2, Time1Flag ;
                retlw   0x00            ;

Try_Venue1Blk   movf    Blk_Type, w     ;
                sublw   Venue1_Blk      ;
                btfss   STATUS, Z       ;
```

```
                goto    Try_Slot2Blk        ;
                movf    Byte_No, w          ;
                movwf   Byte_NoVenue1       ;
                bsf     Flags2, Venue1Flag  ;
                retlw   0x00                ;

Try_Slot2Blk    movf    Blk_Type, w         ;
                sublw   Slot2_Blk           ;
                btfss   STATUS, Z           ;
                goto    Try_Course2Blk      ;
                movf    Byte_No, w          ;
                movwf   Byte_NoSlot2        ;
                bsf     Flags2, Slot2Flag   ;
                bsf     FlagsR, Slot2Flag   ;
                retlw   0x00                ;

Try_Course2Blk  movf    Blk_Type, w         ;
                sublw   Course2_Blk         ;
                btfss   STATUS, Z           ;
                goto    Try_Day2Blk         ;
                movf    Byte_No, w          ;
                movwf   Byte_NoCourse2      ;
                bsf     Flags3, Course2Flag ;
                retlw   0x00                ;

Try_Day2Blk     movf    Blk_Type, w         ;
                sublw   Day2_Blk            ;
                btfss   STATUS, Z           ;
                goto    Try_Time2Blk        ;
                movf    Byte_No, w          ;
                movwf   Byte_NoDay2         ;
                bsf     Flags3, Day2Flag    ;
                retlw   0x00                ;

Try_Time2Blk    movf    Blk_Type, w         ;
                sublw   Time2_Blk           ;
                btfss   STATUS, Z           ;
                goto    Try_Venue2Blk       ;
                movf    Byte_No, w          ;
                movwf   Byte_NoTime2        ;
                bsf     Flags3, Time2Flag   ;
                retlw   0x00                ;

Try_Venue2Blk   movf    Blk_Type, w         ;
                sublw   Venue2_Blk          ;
                btfss   STATUS, Z           ;
                goto    Try_Const1Blk       ;
                movf    Byte_No, w          ;
                movwf   Byte_NoVenue2       ;
                bsf     Flags3, Venue2Flag  ;
                retlw   0x00                ;


Try_Const1Blk   movf    Blk_Type, w         ;
                sublw   Const1_Blk          ;
                btfss   STATUS, Z           ;
                goto    Try_Day3Blk         ;
                movf    Byte_No, w          ;
                movwf   Byte_NoConst1       ;
                bsf     Flags4, Const1Flag  ;
                bsf     FlagsR, 4           ;
                retlw   0x00                ;

Try_Day3Blk     movf    Blk_Type, w         ;
                sublw   Day3_Blk            ;
                btfss   STATUS, Z           ;
                goto    Try_Time3Blk        ;
                movf    Byte_No, w          ;
                movwf   Byte_NoDay3         ;
                bsf     Flags4, Day3Flag    ;
```

```
                retlw   0x00            ;

Try_Time3Blk    movf    Blk_Type, w     ;
                sublw   Time3_Blk       ;
                btfss   STATUS, Z       ;
                goto    Try_Const2Blk   ;
                movf    Byte_No, w      ;
                movwf   Byte_NoTime3    ;
                bsf     Flags4, Time3Flag ;
                retlw   0x00            ;

Try_Const2Blk   movf    Blk_Type, w     ;
                sublw   Const2_Blk      ;
                btfss   STATUS, Z       ;
                goto    Try_Day4Blk     ;
                movf    Byte_No, w      ;
                movwf   Byte_NoConst2   ;
                bsf     Flags4, Const2Flag ;
                bsf     FlagsR, Const2Flag ;
                retlw   0x00            ;

Try_Day4Blk     movf    Blk_Type, w     ;
                sublw   Day4_Blk        ;
                btfss   STATUS, Z       ;
                goto    Try_Time4Blk    ;
                movf    Byte_No, w      ;
                movwf   Byte_NoDay4     ;
                bsf     Flags4, Day4Flag ;
                retlw   0x00            ;

Try_Time4Blk    movf    Blk_Type, w     ;
                sublw   Time4_Blk       ;
                btfss   STATUS, Z       ;
                goto    Try_TestBlk     ;
                movf    Byte_No, w      ;
                movwf   Byte_NoTime4    ;
                bsf     Flags4, Time4Flag ;
                retlw   0x00            ;

Try_TestBlk     movf    Blk_Type, w     ;
                sublw   Test_Blk        ;
                btfss   STATUS, Z       ;
                goto    Try_Course5Blk  ;
                movf    Byte_No, w      ;
                movwf   Byte_NoTest     ;
                bsf     Flags5, TestFlag ;
                bsf     FlagsR, 5       ;
                retlw   0x00            ;

Try_Course5Blk  movf    Blk_Type, w     ;
                sublw   Course5_Blk     ;
                btfss   STATUS, Z       ;
                goto    Try_Day5Blk     ;
                movf    Byte_No, w      ;
                movwf   Byte_NoCourse5  ;
                bsf     Flags5, Course5Flag ;
                 retlw  0x00            ;

Try_Day5Blk     movf    Blk_Type, w     ;
                sublw   Day5_Blk        ;
                btfss   STATUS, Z       ;
                goto    Try_DateBlk     ;
                movf    Byte_No, w      ;
                movwf   Byte_NoDay5     ;
                bsf     Flags5, Day5Flag ;
                retlw   0x00            ;

Try_DateBlk     movf    Blk_Type, w     ;
                sublw   Date_Blk        ;
                btfss   STATUS, Z       ;
```

```
                goto    Try_Time5Blk    ;
                movf    Byte_No, w      ;
                movwf   Byte_NoDate     ;
                bsf     Flags5, DateFlag ;
                retlw   0x00            ;

Try_Time5Blk    movf    Blk_Type, w     ;
                sublw   Time5_Blk       ;
                btfss   STATUS, Z       ;
                goto    Try_Venue5Blk   ;
                movf    Byte_No, w      ;
                movwf   Byte_NoTime5    ;
                bsf     Flags5, Time5Flag ;
                retlw   0x00            ;

Try_Venue5Blk   movf    Blk_Type, w     ;
                sublw   Venue5_Blk      ;
                btfss   STATUS, Z       ;
                goto    Try_AwyMsgBlk   ;
                movf    Byte_No, w      ;
                movwf   Byte_NoVenue5   ;
                bsf     Flags5, Venue5Flag ;
                retlw   0x00            ;

Try_AwyMsgBlk   movf    Blk_Type, w     ;
                sublw   AwyMsg_Blk      ;
                btfss   STATUS, Z       ;
                goto    Try_StopBlk     ;
                movf    Byte_No, w      ;
                movwf   Byte_NoAwyMsg   ;
                bsf     Flags5, AwyMsgFlag ;
                bsf     FlagsR, AwyMsgFlag ;
                retlw   0x00            ;

Try_StopBlk     movf  Blk_Type, w       ;
                sublw  Stop_Blk         ;
                btfss  STATUS, Z        ;
                goto   Try_Error        ;
                movf   Byte_No, w       ;
                sublw  Stop_Blk         ;
                btfss  STATUS, Z        ;
                goto   Try_Error        ;
                bsf    Flags4, Stop_Flag ;
                retlw  0x00             ;

Try_Error nop                           ;
          goto  ISR2                    ;
          return                        ;
;---------------------------------------------------- End ----------------------------------------------------

;-----------------------------ReadIR-----------------------------------

ReadIR    clrwdt
          clrf    Flags
          call    Read_Pulse            ;wait for start pulse (2.4mS)
          btfss   Flags, StartFlag
          goto    ReadIR
          return


Get_Data0       movlw   0x08            ;set up to read 7 bits
                movwf   Bit_Cntr
                clrf    Data_Byte0
Next_RcvBit0    clrwdt
                call    Read_Pulse
                btfsc   Flags,StartFlag ; abort if another Start bit
                goto    ReadIR
                btfsc   Flags,ErrFlag   ; abort if error
                goto    ReadIR
                rlf     Data_Byte0, f
```

```
                bcf     Data_Byte0, 0
                btfss   Flags, Zero
                bsf     Data_Byte0, 0
                decfsz  Bit_Cntr, f
                goto    Next_RcvBit0
                return                          ;wait for start pulse (2.4mS)return

Get_Data1       movlw   0x08                    ;set up to read 7 bits
                movwf   Bit_Cntr
                clrf    Data_Byte1
Next_RcvBit1    clrwdt
                call    Read_Pulse
                btfsc   Flags,StartFlag         ;abort if another Start bit
                goto    ReadIR
                btfsc   Flags,ErrFlag           ;abort if error
                goto    ReadIR
                rlf     Data_Byte1, f
                bcf     Data_Byte1, 0
                btfss   Flags, Zero
                bsf     Data_Byte1, 0
                decfsz  Bit_Cntr, f
                goto    Next_RcvBit1
                return                          ;wait for start pulse (2.4mS)return

Get_Data2       movlw   0x08                    ;set up to read 7 bits
                movwf   Bit_Cntr
                clrf    Data_Byte2
Next_RcvBit2    clrwdt
                call    Read_Pulse
                btfsc   Flags,StartFlag         ;abort if another Start bit
                goto    ReadIR
                btfsc   Flags,ErrFlag           ;abort if error
                goto    ReadIR
                rlf     Data_Byte2, f
                bcf     Data_Byte2, 0
                btfss   Flags, Zero
                bsf     Data_Byte2, 0
                decfsz  Bit_Cntr, f
                goto    Next_RcvBit2
                return                          ;wait for start pulse (2.4mS)return

Get_Data3       movlw   0x08                    ;set up to read 7 bits
                movwf   Bit_Cntr
                clrf    Data_Byte3
Next_RcvBit3    clrwdt
                call    Read_Pulse
                btfsc   Flags,StartFlag         ;abort if another Start bit
                goto    ReadIR
                btfsc   Flags,ErrFlag           ;abort if error
                goto    ReadIR
                rlf     Data_Byte3, f
                bcf     Data_Byte3, 0
                btfss   Flags, Zero
                bsf     Data_Byte3, 0
                decfsz  Bit_Cntr, f
                goto    Next_RcvBit3
                return                          ;wait for start pulse (2.4mS)return

;------------------------- End of ReadIR ---------------------------------

;-------------------------------------------------------------------------
;    Read pulse width, return flag for StartFlag, One, Zero, or ErrFlag
;    output from IR receiver is normally high, and goes low when signal received

Read_Pulse    clrf    LoX

Still_High clrwdt
           btfss      IR_PORT, IR_In      ;and wait until goes low
           goto       Next
```

```
            goto    Still_High

Next        clrwdt
            call    delay           ;waste time to scale pulse
            incf    LoX, f          ;width to 8 bits
            btfss   IR_PORT, IR_In
            goto    Next            ;loop until input high again
            goto    Chk_Pulse

; test if Zero, One, or Start (or error)

Chk_Pulse   clrf    Flags

TryError    movf    LoX, w          ; check if pulse too small
            addlw   d'255' - d'60'  ; if LoX <= 20
            btfsc   STATUS, C
            goto    TryZero
            bsf     Flags, ErrFlag  ; Error found, set flag
            retlw   0x00

TryZero     movf    LoX, w          ; check if zero
            addlw   d'255' - d'80'  ; if LoX <= 60
            btfsc   STATUS, C
            goto    TryOne
            bsf     Flags, Zero     ; Zero found, set flag
            retlw   0x00

TryOne      movf    LoX, w          ; check if one
            addlw   d'255' - d'130' ; if LoX <= 112
            btfsc   STATUS, C
            goto    TryStop
            bsf     Flags, One      ; One found, set flag
            retlw   0x00

TryStop     movf    LoX, w          ; check if start
            addlw   d'255' - d'190' ; if LoX <= 180
            btfsc   STATUS, C
            goto    TryStart
            bsf     Flags, StopFlag ; Start pulse found
            retlw   0x00

TryStart    movf    LoX, w          ; check if start
            addlw   d'255' - d'250' ; if LoX <= 180
            btfsc   STATUS, C
            goto    NoMatch
            bsf     Flags, StartFlag ; Stop pulse found
            retlw   0x00

NoMatch                             ; pulse too long
            bsf     Flags, ErrFlag  ; Error found, set flag
            retlw   0x00

;----------------------------end of pulse measuring routines----------------------------


;----------------InitLCD-------------------
InitLCD     clrwdt
            movlw   LCD_INIT_DDRAM
            call    InstLCD
            movlw   LCD_INIT_CGRAM
            call    InstLCD
            movlw   LCD_FUNCTION_SET
            call    InstLCD
            movlw   LCD_CURSOR
            call    InstLCD
            movlw   LCD_INIT
            call    InstLCD
            movlw   LCD_ENTRY_MODE_SET
            call    InstLCD
            movlw   LCD_CURSOR_HOME
```

```
        call    InstLCD
        movlw   LCD_CLR_DISPLAY
        call    InstLCD
        return
;----------------end InitLCD----------------

;----------------InstLCD--------------------
InstLCD clrwdt
        clrf    PORTD
        bcf     PORTB, RS
        bcf     PORTB, RW
        bcf     PORTB, ENABLE
        movwf   PORTD
        bsf     PORTB, ENABLE
        call    WaitLCD
        bcf     PORTB, ENABLE
        call    WaitLCD         ;
        call    WaitLCD         ;
        call    WaitLCD         ;
        return
;-------------------end----------------------

;-----------------DataLCD-------------------LCD
DataLCD
        clrwdt                          ;
        bsf     PORTB, RS               ; data for LCD
        bcf     PORTB, RW               ;
        bcf     PORTB, ENABLE           ;
        movwf   PORTD                   ;
        nop                             ;
        bsf     PORTB, ENABLE           ;
        call    WaitLCD                 ;
        bcf     PORTB, ENABLE           ;
        retlw   0x00                    ;


;-------------------end-------------------------LCD

;-----------------------------------------------------

ClearLCD        movlw   0x14
                movwf   tmp
_ClearLCD       clrwdt
                movlw   0x20
                call    DataLCD
                call    delay2
                decfsz  tmp, f
                goto    _ClearLCD
                return

;-----------------------------------------------------

;-------------------WaitLCD--------------------LCD
WaitLCD clrwdt
        movlw   WAITLCD         ;
        movwf   waitLCD         ;
_WaitLC call    Wait            ;
        decfsz  waitLCD, 1      ; decrement waitLCD, skip if zero
        goto    _WaitLC         ;
        return                  ; RETURN
;----------------end WaitLCD-----------------LCD

;--------------------Wait---------------------GLOBAL

Wait    clrwdt
        movlw   0xFF            ;
        movwf   w_temp          ;
_Wait   clrwdt                  ; 1 cycle
        decfsz  w_temp, 1       ; 1(2) cycle
        goto    _Wait           ; 2 cycle
```

```
            return                          ; RETURN
;-------------------end Wait-----------------------

;-----------------------------------------------
wait        clrwdt
            movlw     0xFF
            movwf     w_temp2
_wait       clrwdt
            call      Wait
            decfsz    w_temp2,1
            goto      _wait
            return
;-----------------------------------------------
;-----------------------------LCD FUNCTIONS END-----------------------------

delay       movlw     0x08                  ; Delay 10uS
            movwf     tmp1                  ;
            nop                             ;
            nop                             ;
            nop                             ;
_delay      nop                             ;
            decfsz    tmp1, f               ;
            goto      _delay                ;
            return                          ;

delay2      movlw     0xFF                  ; Delay 20mS
            movwf     tmp2                  ;
_delay2     clrwdt                          ;
            call      delay                 ;
            call      delay                 ;
            call      delay                 ;
            call      delay                 ;
            call      delay                 ;
            call      delay                 ;
            call      delay                 ;
            call      delay                 ;
            decfsz    tmp2, f               ;
            goto      _delay2               ;
            return                          ;

delay3      movlw     0x32                  ; Delay 1s
            movwf     tmp3                  ;
_delay3     clrwdt                          ;
            call      delay2                ;
            decfsz    tmp3, 1               ;
            goto      _delay3               ;
            return                          ;

delay4      movlw     0x06                  ; Delay 125mS
            movwf     tmp4                  ;
_delay4     clrwdt                          ;
            call      delay2                ;
            decfsz    tmp4, 1               ;
            goto      _delay4               ;
            return                          ;


;-----------------------------------------------------------
;
;                   NVM Functions Starts
;
;-----------------------------------------------------------

WLect       movlw     0x10                  ; Value for Memory Location
            movwf     w_temp3               ;
            call      WriteNVMG             ; Write at the location
            goto      update                ;

WExt        movlw     0x29                  ; Value for Memory Location
```

```
            movwf   w_temp3         ;
            call    WriteNVMG       ; Write at the location
            goto    update          ;

WSlot1      movlw   0x40            ; In storing data onto memory, it goes the same pattern
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WCourse1    movlw   0x47            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WDay1       movlw   0x50            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WTime1      movlw   0x54            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WVenue1     movlw   0x60            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WSlot2      movlw   0x70            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WCourse2    movlw   0x77            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WDay2       movlw   0x80            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WTime2      movlw   0x84            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WVenue2     movlw   0x90            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WConst1     movlw   0xA0            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;

WDay3       movlw   0xB0            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;


WTime3      movlw   0xB4            ;
            movwf   w_temp3         ;
            call    WriteNVMG       ;
            goto    update          ;
```

```
WConst2 movlw   0xA0            ;
        movwf   w_temp3         ;
        call    WriteNVMG       ;
        goto    update          ;

WDay4   movlw   0xC0            ;
        movwf   w_temp3         ;
        call    WriteNVMG       ;
        goto    update          ;

WTime4  movlw   0xC4            ;
        movwf   w_temp3         ;
        call    WriteNVMG       ;
        goto    update          ;

WTest   movlw   0x00            ;
        movwf   w_temp3         ;
        call    WriteNVMg       ;
        goto    update          ;

WCourse5 movlw  0x05            ;
        movwf   w_temp3         ;
        call    WriteNVMg       ;
        goto    update          ;

WDay5   movlw   0x10            ;
        movwf   w_temp3         ;
        call    WriteNVMg       ;
        goto    update          ;

WDate   movlw   0x14            ;
        movwf   w_temp3         ;
        call    WriteNVMg       ;
        goto    update          ;

WTime5  movlw   0x20            ;
        movwf   w_temp3         ;
        call    WriteNVMg       ;
        goto    update          ;

WVenue5 movlw   0x30            ;
        movwf   w_temp3         ;
        call    WriteNVMg       ;
        goto    update          ;

WAwyMsg         movlw   0x40    ;
                movwf   w_temp3 ;
                call    WriteNVMg
                goto    update  ;

;----------WriteData (Page 1)----------
WriteNVMG
        bcf     PORTC, 3
        call    WaitI2C                 ;
        call    StartP                  ;
        movlw   0xA0                    ; xxxx xxxW
        movwf   byte                    ;----------
        call    OutByte                 ;
        call    GetAckP                 ;
        movf    w_temp3, WREG
        movwf   byte                    ;
        call    OutByte                 ; Send Slave add
        call    GetAckP                 ;
Wr      movf    Blk_No, w
        movwf   tmp
_Wr     clrwdt
        call    GtDtBlk
        call    StData
        decfsz  tmp, 1
        goto    _Wr
```

```
            call        StopP
            bsf         PORTC, 3

            return                                  ;
;-----------------------end-----------------------

;-------------------WriteData (Page 2)--------------------
WriteNVMg
            bcf         PORTC, 3
            call        WaitI2C                     ;
            call        StartP                      ;
            movlw       0xA2                        ; xxxx xxxW
            movwf       byte                        ;----------------------
            call        OutByte                     ;
            call        GetAckP                     ;
            movf        w_temp3, WREG
            movwf       byte                        ;
            call        OutByte                     ; Send Slave add
            call        GetAckP                     ;
wr          movf        Blk_No, w
            movwf       tmp
_wr         clrwdt
            call        GtDtBlk
            call        StData
            decfsz      tmp, 1
            goto        _wr
            call        StopP
            bsf         PORTC, 3

            return                                  ;
;-----------------------end-----------------------

;-----------Get Data Block Function---------
GtDtBlk     ;call       ReadIR
            ;call       Get_Data0
            ;call       delay2

            call        ReadIR
            call        Get_Data0
            call        delay2

            call        ReadIR
            call        Get_Data1
            call        delay2

            call        ReadIR
            call        Get_Data2
            call        delay2

            call        ReadIR
            call        Get_Data3
            call        delay2

            return
;-------------------End-------------------

;---------Storing Data Function----------
StData      movf        Data_Byte0, w
            call        DataIn
            movf        Data_Byte1, w
            call        DataIn
            movf        Data_Byte2, w
            call        DataIn
            movf        Data_Byte3, w
            call        DataIn
            return

DataIn      movwf       byte        ;----------------------
            call        OutByte     ; Send Data
            call        GetAckP     ;----------------------
```

```
            return
;-------------End-----------------------

;-----------------WriteData-------------------
WriteNVM
            call        WaitI2C                 ;
            call        StartP                  ;
            movlw       0xA0                    ; xxxx xxxW
            movwf       byte                    ;--------------------
            call        OutByte                 ;
            call        GetAckP                 ;
            ;movlw      0x00                    ;
            movf        w_temp3, WREG
            movwf       byte                    ;
            call        OutByte                 ; Send Slave add
            call        GetAckP                 ;

            return                              ;
;-----------------------end----------------------

;---------- 'Lecturer' String----------
Lect        movlw       'L'
            call        DataIn
            movlw       'e'
            call        DataIn
            movlw       'c'
            call        DataIn
            movlw       't'
            call        DataIn
            movlw       'u'
            call        DataIn
            movlw       'r'
            call        DataIn
            movlw       'e'
            call        DataIn
            movlw       'r'
            call        DataIn
            movlw       ':'
            call        DataIn
            movlw       ' '
            call        DataIn
            call        StopP
            return
;---------------------------------------

;---------- 'Ext. No:' String---------
Ext         movlw       'E'
            call        DataIn
            movlw       'x'
            call        DataIn
            movlw       't'
            call        DataIn
            movlw       '.'
            call        DataIn
            movlw       0x20
            call        DataIn
            movlw       'N'
            call        DataIn
            movlw       'o'
            call        DataIn
            movlw       ':'
            call        DataIn
            movlw       ' '
            call        DataIn
            call        StopP
            return
;---------------------------------------

;--------'Class Session' String--------
Class       movlw       'C'
```

```
            call        DataIn
            movlw       'l'
            call        DataIn
            movlw       'a'
            call        DataIn
            movlw       's'
            call        DataIn
            movlw       's'
            call        DataIn
            movlw       0x20
            call        DataIn
            movlw       'S'
            call        DataIn
            movlw       'e'
            call        DataIn
            movlw       's'
            call        DataIn
            movlw       's'
            call        DataIn
            movlw       'i'
            call        DataIn
            movlw       'o'
            call        DataIn
            movlw       'n'
            call        DataIn
            movlw       ':'
            call        DataIn
            call        StopP
            return
;-------------------------------------

;------- 'Consultation Hours' String------
Const       movlw       'C'
            call        DataIn
            movlw       'o'
            call        DataIn
            movlw       'n'
            call        DataIn
            movlw       's'
            call        DataIn
            movlw       't'
            call        DataIn
            movlw       ' '
            call        DataIn
            movlw       'H'
            call        DataIn
            movlw       'o'
            call        DataIn
            movlw       'u'
            call        DataIn
            movlw       'r'
            call        DataIn
            movlw       's'
            call        DataIn
            movlw       ':'
            call        DataIn
            call        StopP
            return
;---------------------------------------

;--------- 'Test' String ----------------
Test        movlw       'T'
            call        DataIn
            movlw       'e'
            call        DataIn
            movlw       's'
            call        DataIn
            movlw       't'
            call        DataIn
            movlw       ':'
```

```
            call        DataIn
            call        StopP
            return
;-------------------------------------

;--------------------ReadData(Page 1)--------------------
ReadG
            Movlw       0x01                ;
            Movwf       w_temp              ;
            Call        _loop               ; wait for 100mS

            Call        StartP              ;
            Movlw       0xA0                ; xxxx xxxW
            Movwf       byte                ;----------------------------
            Call        OutByte             ;
            Call        GetAckP             ;
            movf        w_temp3, WREG       ;
            Movwf       byte                ;
            Call        OutByte             ;   Send Slave add
            Call        GetAckP             ;

            Call        StartP              ;
            Movlw       0xA1                ; xxxx xxxR
            Movwf       byte                ;
            call        OutByte             ; supply the slave addr
            call        GetAckP             ;
            call        InByte              ; Read General Data
            call        Display             ;
r           movf        Byte_No, w
            movwf       tmp
_r          clrwdt
            call        read
            decfsz      tmp, 1
            goto        _r
            call        NonAckP             ; send HIGH to the SDA line to indicate NONACK
            call        StopP               ; to tell the device that this is the last data.
            return                          ;
;-----------------------------

;--------------------ReadData(Page 2)--------------------
Readg
            Movlw       0x01                ;
            Movwf       w_temp              ;
            Call        _loop               ; wait for 100mS

            Call        StartP              ;
            Movlw       0xA2                ; xxxx xxxW
            Movwf       byte                ;----------------------------
            Call        OutByte             ;
            Call        GetAckP             ;
            movf        w_temp3, WREG       ;
            Movwf       byte                ;
            Call        OutByte             ;   Send Slave add
            Call        GetAckP             ;

            Call        StartP              ;
            Movlw       0xA3                ; xxxx xxxR
            Movwf       byte                ;
            call        OutByte             ; supply the slave addr
            call        GetAckP             ;
            call        InByte              ; Read General Data
            call        Display             ;
rr          movf        Byte_No, w
            movwf       tmp
_rr         clrwdt
            call        read
            decfsz      tmp, 1
            goto        _rr
            call        NonAckP             ; send HIGH to the SDA line to indicate NONACK
            call        StopP               ; to tell the device that this is the last data.
```

```
        return                  ;
;-------------------------------

read    call    AckP
        call    InByte
        call    delay4
        call    Display
        return

Display movf    byte, WREG
        call    DataLCD
        return
;-------------------------------

ReadL
        Movlw   0x01            ;
        Movwf   w_temp          ;
        Call    _loop           ; wait for 100mS

        Call    StartP          ;
        Movlw   0xA0            ; xxxx xxxW
        Movwf   byte            ;----------------------------
        Call    OutByte         ;
        Call    GetAckP         ;
        Movlw   0x00            ;
        ;movf   w_temp3, WREG
        Movwf   byte            ;
        Call    OutByte         ;   Send Slave add
        Call    GetAckP         ;

        Call    StartP
        Movlw   0xA1            ; xxxx xxxR
        Movwf   byte
        call    OutByte         ; supply the slave addr
        call    GetAckP
        call    InByte          ; Read Name of Lecturer
        call    Display
readL   movlw   0x08            ;
        movwf   tmp             ;
_readL  clrwdt                  ;
        call    read            ;
        decfsz  tmp, f          ;
        goto    _readL          ;
        call    NonAckP         ; send HIGH to the SDA line to indicate NONACK
        call    StopP           ; to tell the device that this is the last data.
        return                  ;


ReadE
        Movlw   0x01            ;
        Movwf   w_temp          ;
        Call    _loop           ; wait for 100mS

        Call    StartP          ;
        Movlw   0xA0            ; xxxx xxxW
        Movwf   byte            ;----------------------------
        Call    OutByte         ;
        Call    GetAckP         ;
        Movlw   0x20            ;
        ;movf   w_temp3, WREG
        Movwf   byte            ;
        Call    OutByte         ;   Send Slave add
        Call    GetAckP         ;

        Call     StartP         ;
        Movlw   0xA1            ; xxxx xxxR
        Movwf   byte            ;
        call    OutByte         ; supply the slave addr
        call    GetAckP         ;
        call    InByte          ; Read Ext No
```

```
        call    Display         ;
readE   movlw   0x0C            ;
        movwf   tmp             ;
_readE  clrwdt                  ;
        call    read            ;
        decfsz  tmp, f          ;
        goto    _readE          ;
        call    NonAckP         ; send HIGH to the SDA line to indicate NONACK
        call    StopP           ; to tell the device that this is the last data.
        return                  ;

ReadC
        Movlw   0x01            ;
        Movwf   w_temp          ;
        Call    _loop           ; wait for 100mS

        Call    StartP          ;
        Movlw   0xA0            ; xxxx xxxW
        Movwf   byte            ;----------------------------
        Call    OutByte         ;
        Call    GetAckP         ;
        Movlw   0x30            ;
        ;movf   w_temp3, WREG
        Movwf   byte            ;
        Call    OutByte         ;   Send Slave add
        Call    GetAckP         ;

        Call    StartP          ;
        Movlw   0xA1            ; xxxx xxxR
        Movwf   byte            ;
        call    OutByte         ; supply the slave addr
        call    GetAckP         ;
        call    InByte          ; Read Class Session
        call    Display         ;
readC   movlw   0x0D            ;
        movwf   tmp             ;
_readC  clrwdt                  ;
        call    read            ;
        decfsz  tmp, f          ;
        goto    _readC          ;
        call    NonAckP         ; send HIGH to the SDA line to indicate NONACK
        call    StopP           ; to tell the device that this is the last data.
        return                  ;

ReadCH
        Movlw   0x01            ;
        Movwf   w_temp          ;
        Call    _loop           ; wait for 100mS

        Call    StartP          ;
        Movlw   0xA0            ; xxxx xxxW
        Movwf   byte            ;----------------------------
        Call    OutByte         ;
        Call    GetAckP         ;
        Movlw   0x70            ;
        ;movf   w_temp3, WREG
        Movwf   byte            ;
        Call    OutByte         ;   Send Slave add
        Call    GetAckP         ;

        Call    StartP          ;
        Movlw   0xA1            ; xxxx xxxR
        Movwf   byte            ;
        call    OutByte         ; supply the slave addr
        call    GetAckP         ;
        call    InByte          ; Read Consultation Hours
        call    Display         ;
readCH  movlw   0x0B            ;
        movwf   tmp             ;
_readCH clrwdt                  ;
```

```
            call        read                ;
            decfsz      tmp, f              ;
            goto        _readCH             ;
            call        NonAckP             ; send HIGH to the SDA line to indicate NONACK
            call        StopP               ; to tell the device that this is the last data.
            return                          ;


;----------------------end----------------------

; 0.2 seconds with wreg = 0x01
;----------Loop for Delay-------------
loop        movwf       w_temp
            bsf         STATUS, C           ; so that if w_temp = 0, after rotate, w_temp= 1
            rlf         w_temp, 1 ; x 2
_loop       call        timer1
            decfsz      w_temp,1            ; here if w_temp == 0, trouble! 255 time looping
            goto        _loop
            return
;-----------------------------------

;----------------------Timer1 (0.1s)--------------------------

setupTimer1             movlw       0x30
                        movwf       T1CON
                        return

timer1                  movlw       0x3C
_timer1                 movwf       TMR1H
                        movlw       0xB0
                        movwf       TMR1L
                        bcf         PIR1,0          ; Clear Timer1 overflow flag
                        bsf         T1CON,0         ; Start Timer1

loopTimer1              clrwdt
                        btfss       PIR1,0
                        goto        loopTimer1
                        bcf         T1CON,0 ; Stop Timer1
                        return

timer2                  movlw       0xFE            ;
                        goto        _timer1         ;
;-------------------------------------------------------------

; i2c_freq
;----------------------WaitI2C------------------------------I2C
WaitI2C
            Clrwdt
            Movlw               0x00
            Movwf               i2c_freq                                    ;
            btfsc               i2c_freq, 7;
            return                                      ;
            btfsc               i2c_freq, 6;
            return                                      ;
            btfsc               i2c_freq, 5;
            return                                      ;
            btfsc               i2c_freq, 4;
            return                                      ;
            btfsc               i2c_freq, 3;
            return                                      ;
            btfsc               i2c_freq, 2;
            return                                      ;
            nop                                         ;
            btfsc               i2c_freq, 1;
            return                                      ;
            nop                                         ;
            nop                                         ;
            btfsc               i2c_freq, 0;
            return                                      ;
            nop                                         ;
```

```
            nop                                     ;
            nop                                     ;
            return                                  ; RETURN
;-------------------------end-------------------------------------

;-------------------------StartP-----------------------------I2C
StartP      ; bit1 = SDA          bit0 = SCL
            Movlw   HIGH_HIGH       ; 1 1 ->
            Movwf   I2C_PORT        ;
            Call    WaitI2C         ;
            Call    WaitI2C         ; 1 1 ->
                                    ;
            movlw   LOW_HIGH        ; 0 1 ->
            movwf   I2C_PORT        ;
            call    WaitI2C         ;
                                    ;
            Movlw   LOW_LOW                 ; 0 0 ->
            Movwf   I2C_PORT        ;
            Call    WaitI2C         ;
            return                  ; RETURN
;-------------------------end-----------------------------I2C
;-------------------------InByte-----------------------------I2C
InByte      movlw   0x09            ;
            Movwf   nbit            ;
_InByte     decf    nbit, 1         ; dec, skip if not zero
            btfsc   STATUS, Z       ;
            return                  ; RETURN...
                                    ; here nbit = 8, 7, 6, 5, 4, 3, 2, 1
            movlw   HIGH_LOW        ; 1 0 0000 1000 : 0 1 -> 0000 0100
            movwf   I2C_PORT        ;
            call    WaitI2C   ;

            movlw   HIGH_HIGH       ; 1 1 -> 0000 1100, release the line high
            movwf   I2C_PORT        ;

            rlf     byte, 1         ; rotate left
            bcf     byte, 0         ;
            call    WaitI2C   ;

            btfsc   I2C_PORT, SDA_IN ; 0000 0000        1000 0000
            bsf     byte, 0         ; here * = '1'
            call    WaitI2C   ;

            movlw   HIGH_LOW        ; 1 0 0000 1000 : 0 1 -> 0000 0100
            movwf   I2C_PORT        ;
            call    WaitI2C   ;
            goto  _InByte           ;
;-------------------------------end-------------------------------I2C
; byte

;-------------------------NonAckP-----------------------------I2C
NonAckP     movlw   HIGH_LOW        ; 1 0 -> 0000 1000 : 0 1 -> 0000 0100
            movwf   I2C_PORT        ;
            call    WaitI2C         ;

            movlw   HIGH_HIGH       ; 1 1 -> 0000 1100
            movwf   I2C_PORT        ;
            call    WaitI2C         ;
            call    WaitI2C         ; 1 1 -> 0000 1100

            movlw   HIGH_LOW        ; 1 0 -> 0000 1000 : 0 1 -> 0000 0100
            movwf   I2C_PORT        ;
            call    WaitI2C         ;
            return                  ; RETURN
;-------------------------------end-------------------------------I2C

;-------------------------------StopP-----------------------------I2C
StopP       movlw   LOW_LOW         ; 0 0 -> 0000 0000
            movwf   I2C_PORT        ;
            call    WaitI2C         ;
```

```
            movlw   LOW_HIGH        ; 0 1 -> 0000 0100 : 1 0 -> 0000 1000
            movwf   I2C_PORT        ;
            call    WaitI2C         ;

            movlw   HIGH_HIGH       ; 1 1 -> 0000 1100
            movwf   I2C_PORT        ;
            call    WaitI2C         ;
            call    WaitI2C         ; 1 1 -> 0000 1100
            return                  ;RETURN
;-------------------------------End-------------------------------I2C

;-------------------------------AckP------------------------------I2C
AckP        movlw   LOW_LOW         ; 0 0 -> 0000 0000
            Movwf   I2C_PORT        ;
            call    WaitI2C         ;

            Movlw   LOW_HIGH        ; 0 1 -> 0000 0100 : 1 0 -> 0000 1000
            Movwf   I2C_PORT        ;
            Call    WaitI2C         ;
            Call    WaitI2C         ; 0 1 -> 0000 0100 : 1 0 -> 0000 1000

            Movlw   LOW_LOW         ; 0 0 -> 0000 0000
            Movwf   I2C_PORT        ;
            Call    WaitI2C         ;
            return                  ; RETURN
;-------------------------------End-------------------------------I2C

;-----------------------------GetAckP-----------------------------I2C
GetAckP     movlw   HIGH_LOW        ; 1 0 -> 0000 0010 : 0 1 -> 0000 0001
            Movwf   I2C_PORT        ;
            call    WaitI2C         ;{1}

            movlw   HIGH_HIGH                   ; 1 1 -> 0000 0011, release the SDA line
            movwf   I2C_PORT                    ; to HIGH and then let the EEPROM decide
                                                        ; to
            call    WaitI2C                     ; {2} pull LOW or release HIGH...
            btfsc   I2C_PORT, SDA_IN ; 0000 0000        1000 0000
            goto    DeviceError                 ; here detect no ACK!
            call    WaitI2C                     ; {3}
            movlw   HIGH_LOW                    ; 1 0 -> 0000 1000 : 0 1 -> 0000 1000
            movwf   I2C_PORT                    ;
            call    WaitI2C                     ; {4}
            return                              ; RETURN
;-------------------------------End-------------------------------I2C

; do something when device is not responding
DeviceError
            bcf     PORTC, 0 ; clearing port c pin 0
RESETMIC
            nop                         ; Looping until the WATCHDOG timer  overflow, this will
            goto    RESETMIC            ; resetting the micon
                                        ;
; pass a byte to 'byte'
;-------------------------------OutByte---------------------------I2C
OutByte             movlw   0x09                ;
                    Movwf   nbit                ; nbit -> control looping
_OutByt             decf    nbit,1
                    btfsc   STATUS, Z           ;
                    return                      ; RETURN...
                                                ; here nbit = 8, 7, 6, 5, 4, 3, 2, 1
                    rlf     byte, 1             ; rotate left, carry
                    btfsc   STATUS, C           ; bit<0> test, skip if clear
                    goto    PHigh               ; here, byte<0> = 1
                                                ; here, byte<0> = 0
                                                ; R3 = SCL, R2 = SDA
PLow        movlw   LOW_LOW                             ; 0 0 ->0000 0000
            Movwf   I2C_PORT                    ;
            Call    WaitI2C                     ;
```

```
        Movlw   LOW_HIGH                ; 0 1 ->0000 0100 : 1 0 ->0000 1000
        Movwf   I2C_PORT                ;
        Call    WaitI2C                 ;
        Call    WaitI2C                 ; 0 1 ->0000 0100 : 1 0 ->0000 1000

        Movlw   LOW_LOW                         ; 0 0 ->0000 0000
        Movwf   I2C_PORT                ;
        Call    WaitI2C                 ;
        Goto    _OutByt                 ;

PHigh   movlw   HIGH_LOW                ; 1 0 ->0000 1000 : 0 1 ->0000 0100
        Movwf   I2C_PORT                ;
        Call    WaitI2C                 ;

        Movlw   HIGH_HIGH               ; 1 1 ->0000 1100
        Movwf   I2C_PORT                ;
        Call    WaitI2C                 ;
        Call    WaitI2C                 ; 1 1 ->0000 1100

        Movlw   HIGH_LOW                ; 1 0 ->0000 1000 : 0 1 ->0000 0100
        Movwf   I2C_PORT                ;
        Call    WaitI2C                 ;
        Goto    _OutByt                 ;
;-------------------------------end OutByte-----------------------------------I2C


end
```

APPENDIX B

Transmitter Code

```
;-------------------------------------------------------------------------------------------------------------------------------
;
;       Programmer: Ahmad Rizal bin Muhammad Arif
;       ID No: 1922
;       Program: Electrical & Electronics Engineering
;       Company: Universiti Teknologi PETRONAS
;
;       Note: This code is the transmitter code. The routine is generating IDC Infra Red protocol.
;             The main routine takes value form a PC then transmit the value
;
;-------------------------------------------------------------------------------------------------------------------------------

#include <P16F84A.inc>

;-------------------------Reserving Register-------------------------------
        cblock  0x20                    ;start of general purpose registers
                count1                  ;used in delay routine
                counta                  ;used in delay routine
                countb
                count
                Delay_Count
                Bit_Cntr
                Data_Byte
                Dev_Byte
                Rcv_Byte
                Pulse
                temp
        endc
;-------------------------------------------------------------------------

;-------------------------------
CONTROL         Equ     PORTA
IR_PORT         Equ     PORTA
IR_Out          Equ     1
C0              Equ     2
C1              Equ     3
;-------------------------------

        org     0x00
        goto    Setup

Setup   bsf     STATUS, RP0     ; Select Bank1
        movlw   0x0C            ;
        movwf   TRISA           ; Select pin 2 & 3 as an input, other as an output
        movlw   0xFF
        movwf   TRISB           ; Select PORTB as an inputs
        bcf     STATUS, RP0
        call    wait3
        goto    main

main    nop
        call    Start_bit       ; Check for Start Bit
        movf    PORTB, w        ; Read from Port B and put in working register
        movwf   Data_Byte       ;
        call    TX_Start        ; Transmit Start Pulse
        call    SendB           ; Transmit Data

        call    Start_bit       ; Check for Start Bit
        movf    PORTB, w        ; Read from Port B and put in working register
        movwf   Data_Byte       ;
        call    TX_Start        ; Transmit Start Pulse
        call    SendB           ; Transmit Data

        goto    main            ;
```

```
;----------Infra Red Routine----------

TX_Start  movlw   d'92'
          call    IR_pulse
          movlw   d'23'
          call    NO_pulse
          retlw   0x00

TX_One    movlw   d'46'
          call    IR_pulse
          movlw   d'23'
          call    NO_pulse
          retlw   0x00

TX_Zero   movlw   d'23'
          call    IR_pulse
          movlw   d'23'
          call    NO_pulse
          retlw   0x00

TX_Stop   movlw   d'69'
          call    IR_pulse
          movlw   d'23'
          call    NO_pulse
          retlw   0x00

IR_pulse
          MOVWF   count           ; Pulses the IR led at 38KHz
irloop    BSF     IR_PORT, IR_Out
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          BCF     IR_PORT, IR_Out
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP
          NOP
          NOP                     ;
          NOP                     ;
          DECFSZ  count,F
          GOTO    irloop
          RETLW   0

NO_pulse
          MOVWF   count           ; Doesn't pulse the IR led
irloop2   BCF     IR_PORT, IR_Out
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
          NOP                     ;
```

```
                BCF     IR_PORT, IR_Out
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP                     ;
                NOP
                NOP
                NOP                     ;
                NOP                     ;
                DECFSZ  count,F
                GOTO    irloop2
                RETLW   0
;----------------------Infra Red Routine End-------------------------

Start_bit   clrwdt
            btfss   CONTROL, C1
            goto    main
            retlw   0x00

Stop_bit    clrwdt
            btfsc   CONTROL, C1
            goto    main
            retlw   0x00

SendB       movlw   0x08
            movwf   temp
_SendB      clrwdt
            btfss   Data_Byte, 7
            goto    sendB
            call    TX_One
            rlf     Data_Byte, 1
_sendB      decfsz  temp, 1
            goto    _SendB
            return

sendB       call    TX_Zero
            rlf     Data_Byte, 1
            goto    _sendB

wait        movlw   0xFF
            movwf   Dev_Byte
_wait       clrwdt
            decfsz  Dev_Byte, f
            goto    _wait
            return

wait2       movlw   0xFF
            movwf   Rcv_Byte
_wait2      clrwdt
            call    wait
            decfsz  Rcv_Byte, f
            goto    _wait2
            return

wait3       movlw   0x12
            movwf   Rcv_Byte
_wait3      clrwdt
            call    wait
            decfsz  Rcv_Byte, f
            goto    _wait3
            return

end
```

APPENDIX C

Software Code

```
/*-------------------------------------------------------------------------------------------------------------------------------
//
//          Programmer: Ahmad Rizal bin Muhammad Arif
//          ID No: 1922
//          Program: Electrical & Electronics Engineering
//          Company: Universiti Teknologi PETRONAS
//
//          Note: This code is the software code
//
//-------------------------------------------------------------------------------------------------------------------------------*/



// IDCParallel4Dlg.cpp : implementation file
//

#include "stdafx.h"
#include "IDCParallel4.h"
#include "IDCParallel4Dlg.h"
#include <conio.h>

#define DATAPORT 0x378 // data printer port
#define CONTROL DATAPORT+2 //        control printer port


#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual void DoDataExchange(CDataExchange* pDX);    // DDX/DDV support
        //}}AFX_VIRTUAL

// Implementation
protected:
        //{{AFX_MSG(CAboutDlg)
        //}}AFX_MSG
        DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
        //{{AFX_DATA_INIT(CAboutDlg)
        //}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
```

```
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIDCParallel4Dlg dialog

CIDCParallel4Dlg::CIDCParallel4Dlg(CWnd* pParent /*=NULL*/)
	: CDialog(CIDCParallel4Dlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CIDCParallel4Dlg)
	m_lecturer = _T("");
	m_extno = _T("");
	m_day1 = _T("");
	m_slot1 = FALSE;
	m_time1 = _T("");
	m_venue1 = _T("");
	m_course1 = _T("");
	m_slot2 = FALSE;
	m_day2 = _T("");
	m_time2 = _T("");
	m_venue2 = _T("");
	m_course2 = _T("");
	m_const1 = FALSE;
	m_const2 = FALSE;
	m_day4 = _T("");
	m_time3 = _T("");
	m_time4 = _T("");
	m_day3 = _T("");
	m_course5 = _T("");
	m_day5 = _T("");
	m_dy = _T("");
	m_venue5 = _T("");
	m_time5 = _T("");
	m_test = FALSE;
	m_month = _T("");
	m_year = _T("");
	m_msg = FALSE;
	m_msgbox = _T("");
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}

void CIDCParallel4Dlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CIDCParallel4Dlg)
	DDX_Text(pDX, IDC_LECTEDIT1, m_lecturer);
	DDV_MaxChars(pDX, m_lecturer, 16);
	DDX_Text(pDX, IDC_EXTNOEDIT2, m_extno);
	DDV_MaxChars(pDX, m_extno, 4);
	DDX_CBString(pDX, IDC_DAY1, m_day1);
	DDV_MaxChars(pDX, m_day1, 8);
	DDX_Check(pDX, IDC_SLOT1, m_slot1);
	DDX_CBString(pDX, IDC_TIME1, m_time1);
	DDX_Text(pDX, IDC_VENUE1, m_venue1);
	DDV_MaxChars(pDX, m_venue1, 16);
	DDX_Text(pDX, IDC_COURSE1, m_course1);
	DDV_MaxChars(pDX, m_course1, 10);
	DDX_Check(pDX, IDC_SLOT2, m_slot2);
	DDX_CBString(pDX, IDC_DAY2, m_day2);
```

```
	DDV_MaxChars(pDX, m_day2, 3);
	DDX_CBString(pDX, IDC_TIME2, m_time2);
	DDX_Text(pDX, IDC_VENUE2, m_venue2);
	DDV_MaxChars(pDX, m_venue2, 16);
	DDX_Text(pDX, IDC_COURSE2, m_course2);
	DDV_MaxChars(pDX, m_course2, 16);
	DDX_Check(pDX, IDC_CONST1, m_const1);
	DDX_Check(pDX, IDC_CONST2, m_const2);
	DDX_CBString(pDX, IDC_DAY4, m_day4);
	DDX_CBString(pDX, IDC_TIME3, m_time3);
	DDX_CBString(pDX, IDC_TIME4, m_time4);
	DDX_CBString(pDX, IDC_DAY3, m_day3);
	DDX_Text(pDX, IDC_COURSE5, m_course5);
	DDX_CBString(pDX, IDC_DAY5, m_day5);
	DDX_CBString(pDX, IDC_DY, m_dy);
	DDX_Text(pDX, IDC_VENUE5, m_venue5);
	DDX_CBString(pDX, IDC_TIME5, m_time5);
	DDX_Check(pDX, IDC_TEST, m_test);
	DDX_CBString(pDX, IDC_MONTH, m_month);
	DDX_CBString(pDX, IDC_YR, m_year);
	DDX_Check(pDX, IDC_MSG, m_msg);
	DDX_Text(pDX, IDC_MSGBOX, m_msgbox);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CIDCParallel4Dlg, CDialog)
	//{{AFX_MSG_MAP(CIDCParallel4Dlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_TRANSMIT, OnTransmit)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CIDCParallel4Dlg message handlers

BOOL CIDCParallel4Dlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING, IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CIDCParallel4Dlg::OnSysCommand(UINT nID, LPARAM lParam)
```

```
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}

// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CIDCParallel4Dlg::OnPaint()
{
	if (IsIconic())
	{
		CPaintDC dc(this); // device context for painting

		SendMessage(WM_ICONERASEBKGND, (WPARAM) dc.GetSafeHdc(), 0);

		// Center icon in client rectangle
		int cxIcon = GetSystemMetrics(SM_CXICON);
		int cyIcon = GetSystemMetrics(SM_CYICON);
		CRect rect;
		GetClientRect(&rect);
		int x = (rect.Width() - cxIcon + 1) / 2;
		int y = (rect.Height() - cyIcon + 1) / 2;

		// Draw the icon
		dc.DrawIcon(x, y, m_hIcon);
	}
	else
	{
		CDialog::OnPaint();
	}
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CIDCParallel4Dlg::OnQueryDragIcon()
{
	return (HCURSOR) m_hIcon;
}

void CIDCParallel4Dlg::IDCData(char data)
{
	_outp(CONTROL,0x00);
	Sleep(5);
	_outp(DATAPORT,data);
	Sleep(10);
	_outp(CONTROL,0x02);
	Sleep(10);

}

void CIDCParallel4Dlg::IDCInfo(int data)
{
	_outp(CONTROL,0x00);
	Sleep(5);
	_outp(DATAPORT,data);
	Sleep(10);
	_outp(CONTROL,0x02);
	Sleep(10);

}
```

```
void CIDCParallel4Dlg::OnTransmit()
{
	// TODO: Add your control notification handler code here
	UpdateData(true);
	IDCInfo(0x00);
	Sleep(1000);

	int lgthlecturer = m_lecturer.GetLength();
	int lgthlectblk = lgthlecturer/4;
	int lgthlectblkR = lgthlecturer % 4;

	int lgthextno = m_extno.GetLength();

	int lgthcourse1 = m_course1.GetLength();
	int lgthcourse1blk = lgthcourse1 / 4;
	int lgthcourse1blkR = lgthcourse1 % 4;

	int lgthday1 = m_day1.GetLength();
	int lgthday1blk = lgthday1 / 4;
	int lgthday1blkR = lgthday1 % 4;

	int lgthtime1 = m_time1.GetLength();
	int lgthtime1blk = lgthtime1 / 4;
	int lgthtime1blkR = lgthtime1 % 4;

	int lgthvenue1 = m_venue1.GetLength();
	int lgthvenue1blk = lgthvenue1 / 4;
	int lgthvenue1blkR = lgthvenue1 % 4;

	int lgthcourse2 = m_course2.GetLength();
	int lgthcourse2blk = lgthcourse2 / 4;
	int lgthcourse2blkR = lgthcourse2 % 4;

	int lgthday2 = m_day2.GetLength();
	int lgthday2blk = lgthday2 / 4;
	int lgthday2blkR = lgthday2 % 4;

	int lgthtime2 = m_time2.GetLength();
	int lgthvenue2 = m_venue2.GetLength();
	int lgthday3 = m_day3.GetLength();
	int lgthtime3 = m_time3.GetLength();

	int lgthday4 = m_day4.GetLength();
	int lgthday4blk = lgthday4 / 4;
	int lgthday4blkR = lgthday4 % 4;

	int lgthtime4 = m_time4.GetLength();
	int lgthtime4blk = lgthtime4 / 4;
	int lgthtime4blkR = lgthtime4 % 4;

	int lgthcourse5 = m_course5.GetLength();
	int lgthcourse5blk = lgthcourse5 / 4;
	int lgthcourse5blkR = lgthcourse5 % 4;

	int lgthday5 = m_day5.GetLength();
	int lgthday5blk = lgthday5 / 4;
	int lgthday5blkR = lgthday5 % 4;

	int lgthdy = m_dy.GetLength();
	int lgthdyblk = lgthdy / 4;
	int lgthdyblkR = lgthdy % 4;

	int lgthmonth = m_month.GetLength();
	int lgthmonthblk = lgthmonth / 4;
	int lgthmonthblkR = lgthmonth % 4;

	int lgthyear = m_year.GetLength();
	int lgthyearblk = lgthyear / 4;
```

```
int lgthyearblkR = lgthyear % 4;

int lgthtime5 = m_time5.GetLength();
int lgthtime5blk = lgthtime5 /4;
int lgthtime5blkR = lgthtime5 % 4;

int lgthvenue5 = m_venue5.GetLength();
int lgthvenue5blk = lgthvenue5 / 4;
int lgthvenue5blkR = lgthvenue5 % 4;

int lgthmsg = m_msgbox.GetLength();
int lgthmsgblk = lgthmsg / 4;
int lgthmsgblkR = lgthmsg % 4;

//Lecturer Name
if (lgthlecturer != 0)
{

        if(lgthlectblkR == 0)
        {
                IDCStBlk(0x00, lgthlectblk, lgthlecturer - 1);


                for(int a=0; a<lgthlecturer; a++)
                {
                        IDCData(m_lecturer[a]);
                        Sleep(20);
                }

                IDCData(0x20);
                Sleep(250);
        }

        if (lgthlectblkR > 0)
        {
                IDCStBlk(0x00, lgthlectblk + 1, lgthlecturer - 1);


                for(int a1=0; a1<lgthlecturer; a1++)
                {
                        IDCData(m_lecturer[a1]);
                        Sleep(20);
                }

                if (lgthlectblkR == 1)
                {
                        for(int a2=0; a2<3; a2++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }

                }

                if (lgthlectblkR == 2)
                {
                        for(int a3=0; a3<2; a3++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if (lgthlectblkR == 3)
                {
                        for(int a4=0; a4<1; a4++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
```

```
                }

                IDCData(0x20);
                Sleep(250);
        }
}

//Sleep(150);

//Ext No:
if(lgthextno == 4)
{

        IDCStBlk(0x01, 0x01, lgthextno - 1);

        for(int b=0; b<lgthextno; b++)
        {
                IDCData(m_extno[b]);
                Sleep(20);
        }

        IDCData(0x20);
        Sleep(250);
}

//Sleep(250);


//Class Session
        //Slot1
if (m_slot1 == 1)
{
        IDCStBlk(2, 2, 6);

        IDCData('S');Sleep(20);IDCData('l');Sleep(20);IDCData('o');Sleep(20);IDCData('t');Sleep(20);
        IDCData(' ');Sleep(20);IDCData('1');Sleep(20);IDCData(':');Sleep(20);IDCData(' ');Sleep(20);
        IDCData(' ');Sleep(750);

        //Course 1
        if(lgthcourse1 != 0)
        {

                if (lgthcourse1blkR == 0)
                {
                        IDCStBlk(3, lgthcourse1blk, lgthcourse1 - 1);

                        for (int c=0; c<lgthcourse1; c++)
                        {
                                IDCData(m_course1[c]);
                                Sleep(20);

                        }

                        IDCData(0x20);
                        Sleep(250);
                }

                if (lgthcourse1blkR > 0)
                {
                        IDCStBlk(3, lgthcourse1blk + 1, lgthcourse1 - 1);

                        for (int c1=0; c1<lgthcourse1; c1++)
                        {
                                IDCData(m_course1[c1]);
                                Sleep(20);
                        }

                        if (lgthcourse1blkR == 1)
                        {
                                for (int c2=0; c2<3; c2++)
```

```
                {
                        IDCData(0x20);
                        Sleep(20);
                }
        }

        if (lgthcourse1blkR == 2)
        {
                for (int c3=0; c3<2; c3++)
                {
                        IDCData(0x20);
                        Sleep(20);
                }
        }

        if (lgthcourse1blkR == 3)
        {
                for(int c4=0; c4<1; c4++)
                {
                        IDCData(0x20);
                        Sleep(20);
                }
        }

        IDCData(0x20);
        Sleep(250);
    }
}

//Sleep(500);

//Day 1
if(lgthday1 != 0)
{
    if(lgthday1blkR == 0)
    {

        IDCStBlk(0x04, lgthday1blk, lgthday1 - 1);

        for (int d=0; d<lgthday1; d++)
        {
                IDCData(m_day1[d]);
                Sleep(20);
        }

        IDCData(0x20);
        Sleep(250);
    }


    if(lgthday1blkR > 0)
    {
        IDCStBlk(0x04, lgthday1blk + 1, lgthday1 - 1);

        for (int d1=0; d1<lgthday1; d1++)
        {
                IDCData(m_day1[d1]);
                Sleep(20);
        }

        if(lgthday1blkR == 1)
        {
                for(int d2=0; d2<3; d2++)
                {
                        IDCData(0x20);
                        Sleep(20);
                }
        }
```

```
                    if(lgthday1blkR == 2)
                    {
                            for(int d3=0; d3<2; d3++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    if(lgthday1blkR == 3)
                    {
                            for(int d4=0; d4<1; d4++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    IDCData(0x20);
                    Sleep(250);
            }
}

//Sleep(500);

//Time1
if(lgthtime1 != 0)
{

            if(lgthtime1blkR == 0)
            {
                    IDCStBlk(0x05, lgthtime1blk, lgthtime1 - 1);

                    for(int e=0; e<lgthtime1; e++)
                    {
                            IDCData(m_time1[e]);
                            Sleep(20);
                    }

                    IDCData(0x20);
                    Sleep(250);
            }

            if(lgthtime1blkR > 0)
            {
                    IDCStBlk(0x05, lgthtime1blk + 1, lgthtime1 - 1);

                    for(int e1=0; e1<lgthtime1; e1++)
                    {
                            IDCData(m_time1[e1]);
                            Sleep(20);
                    }

                    if(lgthtime1blkR == 3)
                    {
                            for(int e2=0; e2<1; e2++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    if(lgthtime1blkR == 2)
                    {
                            for(int e3=0; e3<2; e3++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }
```

```
                    if(lgthtime1blkR == 1)
                    {
                            for(int e4=0; e4<3; e4++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    IDCData(0x20);
                    Sleep(250);
            }


}

//Sleep(500);

//Venue1
if(lgthvenue1 != 0)
{
            if(lgthvenue1blkR == 0)
            {

                    IDCStBlk(0x06, lgthvenue1blk, lgthvenue1 - 1);

                    for(int f=0; f<lgthvenue1; f++)
                    {
                            IDCData(m_venue1[f]);
                            Sleep(20);
                    }

                    IDCData(0x20);
                    Sleep(250);
            }

            if(lgthvenue1blkR > 0)
            {

                    IDCStBlk(0x06, lgthvenue1blk + 1, lgthvenue1 - 1);

                    for(int f1=0; f1<lgthvenue1; f1++)
                    {
                            IDCData(m_venue1[f1]);
                            Sleep(20);
                    }

                    if(lgthvenue1blkR == 3)
                    {
                            for(int f2=0; f2<1; f2++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    if(lgthvenue1blkR == 2)
                    {
                            for(int f3=0; f3<2; f3++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    if(lgthvenue1blkR == 1)
                    {
                            for(int f4=0; f4<3; f4++)
```

```
                                        {
                                                IDCData(0x20);
                                                Sleep(20);
                                        }
                                }

                                IDCData(0x20);
                                Sleep(250);
                        }
                }
}

//Sleep(500);

//Class Session
        //Slot 2
        if(m_slot2 == 1)
        {
                IDCStBlk(7, 2, 6);

                IDCData('S');Sleep(20);IDCData('l');Sleep(20);IDCData('o');Sleep(20);IDCData('t');Sleep(20);
                IDCData(' ');Sleep(20);IDCData('2');Sleep(20);IDCData(':');Sleep(20);IDCData(' ');Sleep(20);
                IDCData(' ');Sleep(750);

                //Course2
                if(lgthcourse2 != 0)
                {

                        if (lgthcourse2blkR == 0)
                        {
                                IDCStBlk(8, lgthcourse2blk, lgthcourse2 - 1);

                                for (int g=0; g<lgthcourse2; g++)
                                {
                                        IDCData(m_course2[g]);
                                        Sleep(20);
                                }

                                IDCData(0x20);
                                Sleep(250);
                        }

                        if (lgthcourse2blkR > 0)
                        {
                                IDCStBlk(8, lgthcourse2blk + 1, lgthcourse2 - 1);

                                for (int g=0; g<lgthcourse2; g++)
                                {
                                        IDCData(m_course2[g]);
                                        Sleep(20);
                                }

                                if (lgthcourse2blkR == 3)
                                {
                                        for (int g=0; g<1; g++)
                                        {
                                                IDCData(0x20);
                                                Sleep(20);
                                        }
                                }

                                if (lgthcourse2blkR == 2)
                                {
                                        for (int g=0; g<2; g++)
                                        {
                                                IDCData(0x20);
                                                Sleep(20);
                                        }
                                }
```

```
                    if (lgthcourse2blkR == 1)
                    {
                            for(int g=0; g<3; g++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    IDCData(0x20);
                    Sleep(250);
            }
}
//Sleep(500);

//Day2
if(lgthday2 != 0)
{

            if(lgthday2blkR == 0)
            {
                    IDCStBlk(9, lgthday2blk, lgthday2 - 1);

                    for(int h=0; h<lgthday2; h++)
                    {
                            IDCData(m_day2[h]);
                            Sleep(20);
                    }

                    IDCData(0x20);
                    Sleep(250);
            }

            if(lgthday2blkR > 0)
            {
                    IDCStBlk(9, lgthday2blk + 1, lgthday2 - 1);

                    for(int h=0; h<lgthday2; h++)
                    {
                            IDCData(m_day2[h]);
                            Sleep(20);
                    }

                    if(lgthday2blkR == 3)
                    {
                            for(int h=0; h<1; h++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    if(lgthday2blkR == 2)
                    {
                            for(int h=0; h<2; h++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }

                    if(lgthday2blkR == 1)
                    {
                            for(int h=0; h<3; h++)
                            {
                                    IDCData(0x20);
                                    Sleep(20);
                            }
                    }
```

```
                IDCData(0x20);
                Sleep(250);
        }
}

//Sleep(500);
//Time2
if(lgthtime2 != 0)
{
        int lgthtime2blk = lgthtime2 / 4;
        int lgthtime2blkR = lgthtime2 % 4;

        if(lgthtime2blk == 0)
        {
                IDCStBlk(0x0A, lgthtime2blk, lgthtime2 - 1);

                for(int i=0; i<lgthtime2; i++)
                {
                        IDCData(m_time2[i]);
                        Sleep(20);
                }

                IDCData(0x20);
                Sleep(250);
        }

        if(lgthtime2blkR > 0)
        {
                IDCStBlk(0x0A, lgthtime2blk + 1, lgthtime2 - 1);

                for(int i=0; i<lgthtime2; i++)
                {
                        IDCData(m_time2[i]);
                        Sleep(20);
                }

                if(lgthtime2blkR == 3)
                {
                        for(int i=0; i<1; i++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthtime2blkR == 2)
                {
                        for(int i=0; i<2; i++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthtime2blkR == 1)
                {
                        for(int i=0; i<3; i++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                IDCData(0x20);
                Sleep(250);
        }
}

//Sleep(500);
//Venue2*/
```

```
        if(lgthvenue2 != 0)
        {
                int lgthvenue2blk = lgthvenue2 / 4;
                int lgthvenue2blkR = lgthvenue2 % 4;

                if(lgthvenue2blkR == 0)
                {
                        IDCStBlk(0x0B, lgthvenue2blk, lgthvenue2 -1);

                        for(int j=0; j<lgthvenue2; j++)
                        {
                                IDCData(m_venue2[j]);
                                Sleep(20);
                        }

                        IDCData(0x20);
                        Sleep(250);
                }

                if(lgthvenue2blkR > 0)
                {
                        IDCStBlk(0x0B, lgthvenue2blk + 1, lgthvenue2 - 1);

                        for(int j=0; j<lgthvenue2; j++)
                        {
                                IDCData(m_venue2[j]);
                                Sleep(20);
                        }

                        if(lgthvenue2blkR ==3)
                        {
                                for(int j=0; j<1; j++)
                                {
                                        IDCData(0x20);
                                        Sleep(20);
                                }
                        }

                        if(lgthvenue2blkR == 2)
                        {
                                for(int j=0; j<2; j++)
                                {
                                        IDCData(0x20);
                                        Sleep(20);
                                }
                        }

                        if(lgthvenue2blkR == 1)
                        {
                                for(int j=0; j<3; j++)
                                {
                                        IDCData(0x20);
                                        Sleep(20);
                                }
                        }

                        IDCData(0x20);
                        Sleep(250);
                }
        }
}

//Sleep(500);
//Consultation Hour

if(m_const1 == 1)
{
        IDCStBlk(0x0C, 3, 0x0B);
```

```
IDCData('C'); Sleep(20); IDCData('o'); Sleep(20); IDCData('n'); Sleep(20); IDCData('s');
Sleep(20);
IDCData('t'); Sleep(20); IDCData('.'); Sleep(20); IDCData(' '); Sleep(20); IDCData('H'); Sleep(20);
IDCData('o'); Sleep(20); IDCData('u'); Sleep(20); IDCData('r'); Sleep(20); IDCData(':');
Sleep(20);
IDCData(0x20); Sleep(750);

//Day3
if(lgthday3 != 0)
{
        int lgthday3blk = lgthday3 / 4;
        int lgthday3blkR = lgthday3 % 4;

        if(lgthday3blkR == 0)
        {
                IDCStBlk(0x0D, lgthday3blk, lgthday3 - 1);

                for(int k=0; k<lgthday3; k++)
                {
                        IDCData(m_day3[k]);
                        Sleep(20);
                }

                IDCData(0x20);
                Sleep(250);
        }

        if(lgthday3blkR > 0)
        {
                IDCStBlk(0x0D, lgthday3blk + 1, lgthday3 - 1);

                for(int k1=0; k1<lgthday3; k1++)
                {
                        IDCData(m_day3[k1]);
                        Sleep(20);
                }

                if(lgthday3blkR == 3)
                {
                        for(int k=0; k<1; k++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthday3blkR == 2)
                {
                        for(int k=0; k<2; k++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthday3blkR == 1)
                {
                        for(int k=0; k<3; k++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                IDCData(0x20);
                Sleep(250);
        }
}

//Sleep(500);
```

```
		//Time3
		if(lgthtime3 != 0)
		{
			int lgthtime3blk = lgthtime3 / 4;
			int lgthtime3blkR = lgthtime3 % 4;

			if(lgthtime3blkR == 0)
			{
				IDCStBlk(0x0E, lgthtime3blk, lgthtime3 - 1);

				for(int l=0; l<lgthtime3; l++)
				{
					IDCData(m_time3[l]);
					Sleep(20);
				}

				IDCData(0x20);
				Sleep(20);
			}

			if(lgthtime3blkR > 0)
			{
				IDCStBlk(0x0E, lgthtime3blk + 1, lgthtime3 - 1);

				for(int l=0; l<lgthtime3; l++)
				{
					IDCData(m_time3[l]);
					Sleep(20);
				}

				if(lgthtime3blkR == 3)
				{
					for(int l=0; l<1; l++)
					{
						IDCData(0x20);
						Sleep(20);
					}
				}

				if(lgthtime3blkR == 2)
				{
					for(int l=0; l<2; l++)
					{
						IDCData(0x20);
						Sleep(20);
					}
				}

				if(lgthtime3blkR == 1)
				{
					for(int l=0; l<3; l++)
					{
						IDCData(0x20);
						Sleep(20);
					}
				}

				IDCData(0x20);
				Sleep(250);
			}
		}
	}

	if(m_const2 == 1)
	{
		//Consultation Hour 2
		Sleep(500);
		IDCStBlk(0x0F, 3, 0x0B);
```

```
IDCData('C'); Sleep(20); IDCData('o'); Sleep(20); IDCData('n'); Sleep(20); IDCData('s');
Sleep(20);
IDCData('t'); Sleep(20); IDCData('.'); Sleep(20); IDCData(' '); Sleep(20); IDCData('H'); Sleep(20);
IDCData('o'); Sleep(20); IDCData('u'); Sleep(20); IDCData('r'); Sleep(20); IDCData(':');
Sleep(20);
IDCData(0x20); Sleep(750);

//Day4
if(lgthday4 != 0)
{
        if(lgthday4blkR == 0)
        {
                IDCStBlk(0x10, lgthday4blk, lgthday4 - 1);

                for(int m=0; m<lgthday4; m++)
                {
                        IDCData(m_day4[m]);
                        Sleep(20);
                }

                IDCData(0x20);
                Sleep(250);
        }

        if(lgthday4blkR > 0)
        {
                IDCStBlk(0x10, lgthday4blk + 1, lgthday4 - 1);

                for(int m=0; m<lgthday4; m++)
                {
                        IDCData(m_day4[m]);
                        Sleep(20);
                }

                if(lgthday4blkR == 3)
                {
                        for(m=0; m<1; m++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthday4blkR == 2)
                {
                        for(m=0; m<2; m++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthday4blkR == 1)
                {
                        for(m=0; m<3; m++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                IDCData(0x20);
                Sleep(250);
        }
}

//Time4
//Sleep(500);
if(lgthtime4 != 0)
```

```
{
    if(lgthtime4blkR == 0)
    {
        IDCStBlk(0x11, lgthtime4blk, lgthtime4 - 1);

        for(int m=0; m<lgthtime4; m++)
        {
            IDCData(m_time4[m]);
            Sleep(20);
        }

        IDCData(0x20);
        Sleep(250);
    }

    if(lgthtime4blkR > 0)
    {
        IDCStBlk(0x11, lgthtime4blk + 1, lgthtime4 - 1);

        for(int m=0; m<lgthtime4; m++)
        {
            IDCData(m_time4[m]);
            Sleep(20);
        }

        if(lgthtime4blkR == 3)
        {
            for(int m=0; m<1; m++)
            {
                IDCData(0x20);
                Sleep(20);
            }
        }

        if(lgthtime4blkR == 2)
        {
            for(int m=0; m<2; m++)
            {
                IDCData(0x20);
                Sleep(20);
            }
        }

        if(lgthtime4blkR == 1)
        {
            for(int m=0; m<3; m++)
            {
                IDCData(0x20);
                Sleep(20);
            }
        }

        IDCData(0x20);
        Sleep(250);
    }
    }
}

//Test
if(m_test == 1)
{
    IDCStBlk(0x12, 0x02, 0x04);

    IDCData('T'); Sleep(20); IDCData('e'); Sleep(20); IDCData('s'); Sleep(20); IDCData('t');
Sleep(20);
    IDCData(':'); Sleep(20); IDCData(' '); Sleep(20); IDCData(' '); Sleep(20); IDCData(' '); Sleep(20);
    IDCData(' '); Sleep(750);

    if(lgthcourse5 != 0)
    {
```

```
		if(lgthcourse5blkR == 0)
		{
			IDCStBlk(0x13, lgthcourse5blk, lgthcourse5 - 1);

			for(int n=0; n<lgthcourse5; n++)
			{
				IDCData(m_course5[n]);
				Sleep(20);
			}

			IDCData(0x20);
			Sleep(250);
		}

		if(lgthcourse5blkR > 0)
		{
			IDCStBlk(0x13, lgthcourse5blk + 1, lgthcourse5 - 1);

			for(int n=0; n<lgthcourse5; n++)
			{
				IDCData(m_course5[n]);
				Sleep(20);
			}

			if(lgthcourse5blkR == 3)
			{
				for(int n=0; n<1; n++)
				{
					IDCData(0x20);
					Sleep(20);
				}
			}

			if(lgthcourse5blkR == 2)
			{
				for(int n=0; n<2; n++)
				{
					IDCData(0x20);
					Sleep(20);
				}
			}

			if(lgthcourse5blkR == 1)
			{
				for(int n=0; n<3; n++)
				{
					IDCData(0x20);
					Sleep(20);
				}
			}

			IDCData(0x20);
			Sleep(250);
		}
	}

	//Sleep(500);

	//Day5
	if(lgthday5 != 0)
	{
		if(lgthday5blkR == 0)
		{
			IDCStBlk(0x14, lgthday5blk, lgthday5 - 1);

			for(int o=0; o<lgthday5; o++)
			{
				IDCData(m_day5[o]);
				Sleep(20);
			}
```

```
                IDCData(0x20);
                Sleep(250);
        }

        if(lgthday5blkR > 0)
        {
                IDCStBlk(0x14, lgthday5blk + 1, lgthday5 - 1);

                for(int o=0; o<lgthday5; o++)
                {
                        IDCData(m_day5[o]);
                        Sleep(20);
                }

                if(lgthday5blkR == 3)
                {
                        for(int o=0; o<1; o++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthday5blkR == 2)
                {
                        for(int o=0; o<2; o++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthday5blkR == 1)
                {
                        for(int o=0; o<3; o++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                IDCData(0x20);
                Sleep(250);
        }
}

//Date
//Sleep(500);
if(lgthdy != 0 && lgthmonth != 0 && lgthyear != 0)
{
        IDCStBlk(0x15, 0x02, 0x07);

        IDCData(m_dy[0]); Sleep(20); IDCData(m_dy[1]); Sleep(20);
        IDCData('/'); Sleep(20);
        IDCData(m_month[0]); Sleep(20); IDCData(m_month[1]); Sleep(20);
        IDCData('/'); Sleep(20);
        IDCData(m_year[0]); Sleep(20); IDCData(m_year[1]); Sleep(20);
        IDCData(0x20); Sleep(250);
}

//Time5
//Sleep(500);
if(lgthtime5 != 0)
{
        if(lgthtime5blkR == 0)
        {
                IDCStBlk(0x16, lgthtime5blk, lgthtime5 - 1);

                for(int p=0; p<lgthtime5; p++)
```

```
                {
                        IDCData(m_time5[p]);
                        Sleep(20);
                }

                IDCData(0x20);
                Sleep(20);
        }

        if(lgthtime5blkR > 0)
        {
                IDCStBlk(0x16, lgthtime5blk + 1, lgthtime5 - 1);

                for(int p=0; p<lgthtime5; p++)
                {
                        IDCData(m_time5[p]);
                        Sleep(20);
                }

                if(lgthtime5blkR == 3)
                {
                        for(int p=0; p<1; p++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthtime5blkR == 2)
                {
                        for(int p=0; p<2; p++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                if(lgthtime5blkR == 1)
                {
                        for(int p=0; p<3; p++)
                        {
                                IDCData(0x20);
                                Sleep(20);
                        }
                }

                IDCData(0x20);
                Sleep(250);
        }
}

//Venue5
//Sleep(500);
if(lgthvenue5 != 0)
{
        if(lgthvenue5blkR == 0)
        {
                IDCStBlk(0x17, lgthvenue5blk, lgthvenue5 - 1);

                for(int q=0; q<lgthvenue5; q++)
                {
                        IDCData(m_venue5[q]);
                        Sleep(20);
                }

                IDCData(0x20);
                Sleep(20);
        }

        if(lgthvenue5blkR > 0)
```

```
                {
                        IDCStBlk(0x17, lgthvenue5blk + 1, lgthvenue5 - 1);

                        for(int q=0; q<lgthvenue5; q++)
                        {
                                IDCData(m_venue5[q]);
                                Sleep(20);
                        }

                        if(lgthvenue5blkR == 3)
                        {
                                for(int q=0; q<1; q++)
                                {
                                        IDCData(0x20);
                                        Sleep(20);
                                }
                        }

                        if(lgthvenue5blkR == 2)
                        {
                                for(int q=0; q<2; q++)
                                {
                                        IDCData(0x20);
                                        Sleep(20);
                                }
                        }

                        if(lgthvenue5blkR == 1)
                        {
                                for(int q=0; q<3; q++)
                                {
                                        IDCData(0x20);
                                        Sleep(20);
                                }
                        }

                        IDCData(0x20);
                        Sleep(250);
                }
        }
}

//Away Message
//Sleep(500);
if(m_msg == 1)
{
        if(lgthmsg != 0)
        {
                if(lgthmsgblkR == 0)
                {
                        IDCStBlk(0x18, lgthmsgblk, lgthmsg - 1);

                        for(int r=0; r<lgthmsg; r++)
                        {
                                IDCData(m_msgbox[r]);
                                Sleep(20);
                        }

                        IDCData(0x20);
                        Sleep(20);
                }

                if(lgthmsgblkR > 0)
                {
                        IDCStBlk(0x18, lgthmsgblk + 1, lgthmsg - 1);

                        for(int r=0; r<lgthmsg; r++)
                        {
                                IDCData(m_msgbox[r]);
                                Sleep(20);
```

```
                                        }

                                        if(lgthmsgblkR == 3)
                                        {
                                                for(int r=0; r<1; r++)
                                                {
                                                        IDCData(0x20);
                                                        Sleep(20);
                                                }
                                        }

                                        if(lgthmsgblkR == 2)
                                        {
                                                for(int r=0; r<2; r++)
                                                {
                                                        IDCData(0x20);
                                                        Sleep(20);
                                                }
                                        }

                                        if(lgthmsgblkR == 1)
                                        {
                                                for(int r=0; r<3; r++)
                                                {
                                                        IDCData(0x20);
                                                        Sleep(20);
                                                }
                                        }

                                        IDCData(0x20);
                                        Sleep(250);
                                }
                        }
                }

                Sleep(250);

                IDCStBlk(0xAA, 0x00, 0xAA);
                IDCData(0x20);
                Sleep(50);
                _outp(CONTROL, 0x03);
                Sleep(100);
                _outp(CONTROL, 0x02);
}


void CIDCParallel4Dlg::IDCStBlk(int Blk_Type, int Blk_No, int Byte_No)
{
        IDCInfo(Blk_Type);
        Sleep(20);
        IDCInfo(Blk_No);
        Sleep(20);
        IDCInfo(Byte_No);
        Sleep(250);

}
```