Nguyen Quang Vu_1005157

Report: Project on an Echo Device

Helsinki Metropolia University of Applied Sciences

Embedded Systems Programing Course TI00AA35-2002

18 May 2012 03:33

Abstract

Author(s) Title Number of Pages Date	Nguyen Quang Vu_1005157 Project on an Echo Device 22 pages 18 May 2012 03:33
Course	Embedded Systems Programing Course TI00AA35-2002
Degree Programme	Information Technology
Specialisation option	Embedded Systems Engineering
Instructor	Kimmo Sauren
Keywords	Echo device, UART, serial communication, RS-232,…

Contents

1 Introduction

With an increasing popularity of the PSoC (Programmable Systems on Chip), people nowadays have turned to this chip. The Embedded Systems Programming course TI00AA35-2002 offers the teaching of the PSoC, and at the end of the course, a final project is given to the students who are interested in the Embedded Systems Engieering field. Among several topics for the final project, the topic of devising an echo device using the PSoC has been chosen.

The aim of this project was to create an echo device using PSoC which can communicate with a personal computer via serial communication. In particular, using a terminal from the PC, we input text from the keyboard of the PC and then it will be displayed on the LCD display. Also, since this is an echo device, text is echoed back to the terminal via serial connection (RS-232 cable).

This report presents the design of an echo device using the chip PSoC1 CY8C27443-24PVXI PDIP. An RS-232 cable is used for serial connection between a PC and the PSoC circuit board. Besides, a simple circuit for translating the ±10 V RS232 signals to transistor-transistor logic (TTL) level signals of the PSoC is used.

A detailed description of the design of the hardware and software are given. Following these are explanations on how the program works and how the software is tested. Finally, a conclusion on the project will be drawn to give some suggestions, limitations as well as useful experiences.

2 Technical measures

The following gives a list of components and parts needed to complete the echo device project.

- PSoC Designer 5.2 software
- PSoC Mini Programmer
- USB Cable

- A 9-pin female serial cable (DB-9) for serial connection between the echo device (PSoC) and the PC
- PSoC platform with LCD display and CY8C27443-24PVXI PDIP chip included. The PSoC printed circuit board is designed by Juho Vesanen; more information about the PSoC platform can be found from the webpage : http://prj.fi/221/

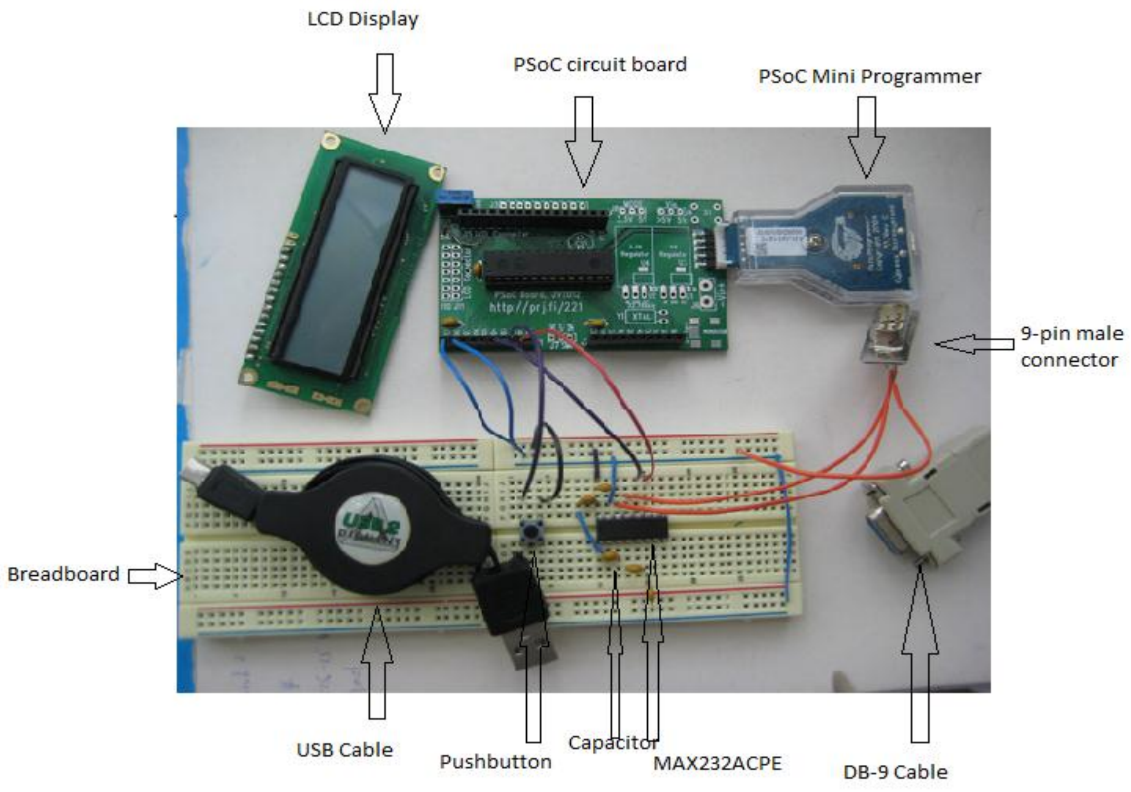


Figure 1. Technical measures for the echo device project

- A PCB pushbutton for user to choose Online/Offline mode of the echo device
- A circuit for getting 3.3 to 5V TTL device to talk to the PC with an RS-232 serial port using MAX232ACPE, and five 0.1µF ceramic capacitors
- A breadboard for building the circuit using MAX232ACPE.
- A 9-pin male (DB-9) connector
- Wires for connecting circuit components

3 Hardware design

This chapter will give detailed information about the design with respect to the hardware of the project. User modules and placement, parameter setting, global resources, pin configuration, hardware connections are described.

3.1 User modules and placement

User modules	Placements
UART	DCB02 (Tx configuration) DCB03 (Rx configuration)
LCD	Port_2 (pins from 0 to 6)

Table 1. User modules and their placement

The user modules UART and LCD are used in the project. the UART is used as both a serial transmitter (Tx) and a receiver (Rx). The TX PSoC block gives transmitter functionality and the RX PSoC block gives receiver functionality.

The LCD is used for display text inputted from the keyboard of the PC.

3.2 User modules and parameter settings

The following tables give parameter settings for the UART user modules and the LCD. An explanation on the setting is also included in the tables.

UART		
Parameter	Value	Explanations
Name	UART	
User module	UART	
Clock	VC3	Input is 153.8 KHz clock (8 times baud rate - 19200). VC3 = SysClk / 156
RX Input	Row_0_Input_0	Routed from pin P0[4] through GlobalInEven_4
TX Output	Row_0_Output_1	Routed to pin P0[1] through GlobalOutEven_1
UART		
Parameter	Value	Explanations

TX Interrupt Mode	TXRegEmpty	Not used
ClockSync	Sync to SysClk	Clock is synchronized with the SysClk, as VC3 is a derivative of SysClk
RxCmdBuffer	Enable	Enables Command buffer, and stores the received command in a RAM buffer
RxBufferSize	33	Bytes length of buffer is 33 characters
CommandTerminator	13	Carriage return (13) is the command terminator
Param_Delimiter	32	Space (32) is the parameter delimiter, but this parameter is not used for the project
IgnoreCharsBelow	32	Ignore control characters that have ASCII value below 32
Enable_BackSpace	Disable	Not used
RX Output	None	Not used
RX Clock Out	None	Not used
TX Clock Out	None	Not used
InvertRX Input	Normal	Do not invert Rx Input

Table 2. UART user module and its parameter setting

LCD		
Parameter	Value	Explanations
Name	LCD	
User Module	LCD	
Version	1.60	
LCDPort	Port_2	Port2 (pins from 0 to 6) is used to connect the LCD to the PSoC platform
BarGraph	Disable	Not used

Table 3. LCD user module and its parameter setting

Our desired baud rate is 19200 bits/s, and the Clock for the UART user module must be 8 times the desired baud rate. That's why the source clock for the UART is VC3. The way of setting VC3 is shown in the Global Sources part.

3.3 Global resources

Important Global Resources		
Parameter	Value	Explanations
CPU_Clock	12MHz (SysClk / 2)	Sets the CPU frequency to 12 MHz
VC3 Source	SysClk / 1	Set System Clock as the source for VC3
VC3 Divider	156	Divide 24 MHz system clock by 156 (effective baud rate is 19.2 kbps)

Table 4. Global resources

The table above shows only parameters of the Global Resources necessary for the project. Other parameters are set as default.

3.4 Pin configuration

Now, the way of configuring the pins are shown in the following table.

Pin	Name	Select	Drive	Interrupt
P0[1]	Port_0_1	globalOutEven_1	Strong	DisableInt
P0[2]	statusButton	StdCPU	Pull Down	RisigEdge
P0[4]	Port_0_4	GlobalInEven_4	High Z	DisableInt

Table 5. Pin configuration for the project

One thing to keep in mind about the PSoC platform is that for P0[0], it is designed as reset. Therefore, this pin P0[0] is never used in any circumstances except for reset mode. Right next to pin P0[7] is the GND pin, and next to this GND pin is the VCC (5V) pin. The 2 pins, GND and VCC, are used to support ground and to supply power to the circuit for getting 3.3 to 5V TTL device to talk to the PC with an RS-232 serial port using MAX232ACPE.

3.5 Hardware connections

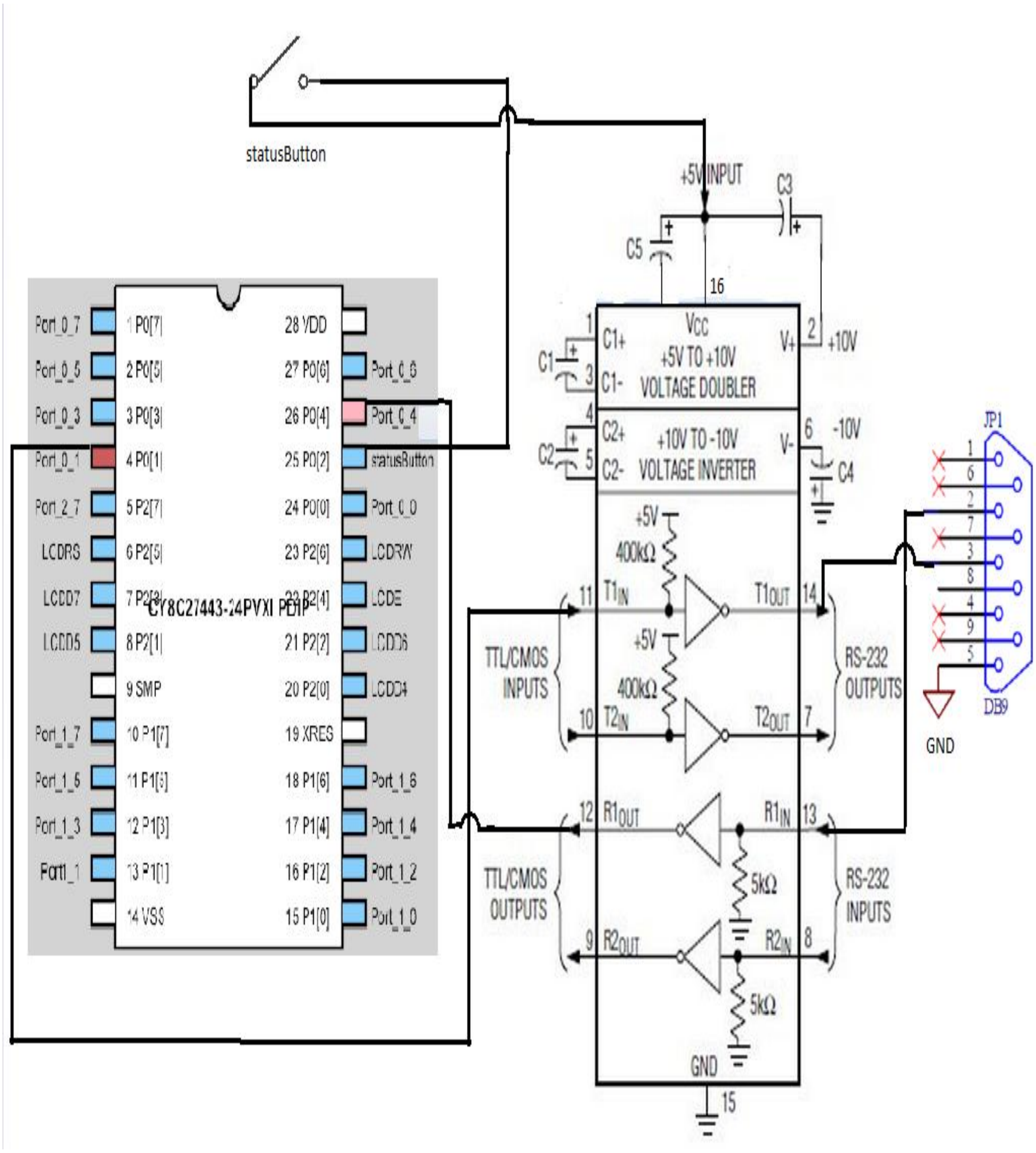


Figure 2. Schematic diagram

The schematic diagram shows how the circuit is wired up for the echo device project. In the middle of the diagram is the MAX232ACPE chip which is used to get 3.3 to 5V TTL device to talk to the PC with an RS-232 serial port. All the capacitors connect to the MAX232ACPE have a value of 0.1 μF. The capacitors used in the project are not polarized, so the terminals of the capacitors can be in either the direction. The MAX232ACPE is powered to 5V from the breadboard.

Pins 2, 3, and 5 of the DB-9 male connector are connected to the MAX232ACPE chip. To make it clear, pin 5 of the DB-9 connector is connected to the ground of the bread-

board. Pin 2 and pin 3 of the DB-9 connector are connected to pin 13 and pin 14 of MAX232ACPE chip, respectively. All other pins of the DB-9 male connector are left unused.
Pins VCC (5V) and GND of the PSoC platform are connected to the VCC and GND of the breadboard. Also, pins P0[1] and P0[4] of the PSoC platform are connected to pin 11 and pin 12 of the MAX232ACPE chip, respectively.
Finally, a PCB pushbutton is used for selecting online/offline mode of the project. One pin of the pushbutton is wired up to the VCC of the breadboard. Meanwhile, the other pin is connected to pin P0[2] of the PSoC platform.

4 Software design

The design of software began with the algorithm for the project. After having ideas about the algorithm, a flowchart is drawn to show how the coding is performed. The following flowchart shows how the coding is done and also the execution of the program. There are three types of flowchart in the process of software design. The first flowchart is the flowchart for main() function. The second one is flowchart for subroutines. And finally, it is the flowchart for interrupt service routine.

First the flowchart for the main function is shown. Flowing this is the flowchart for an interrupt service routine. Lastly, flowcharts for subroutines are given.

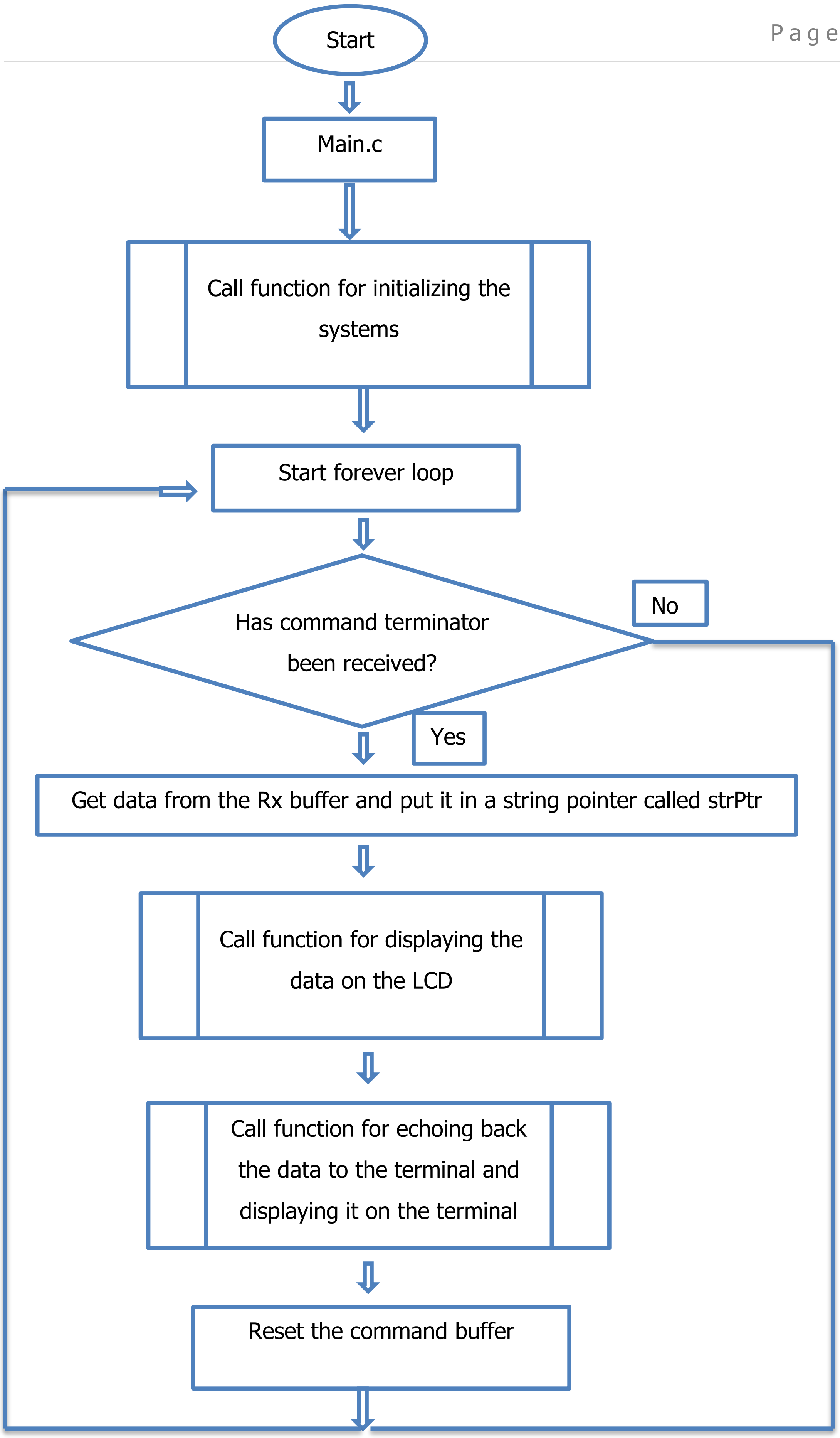

Start
Main.c
Call function for initializing the systems
Start forever loop
Has command terminator been received?
No
Yes
Get data from the Rx buffer and put it in a string pointer called strPtr
Call function for displaying the data on the LCD
Call function for echoing back the data to the terminal and displaying it on the terminal
Reset the command buffer

Figure 3. Flowchart for main function

When the pushbutton is pressed, the program will be interrupted immediately and then jump to the interrupt service routine. The following is a flowchart for the interrupt service routine of the button.

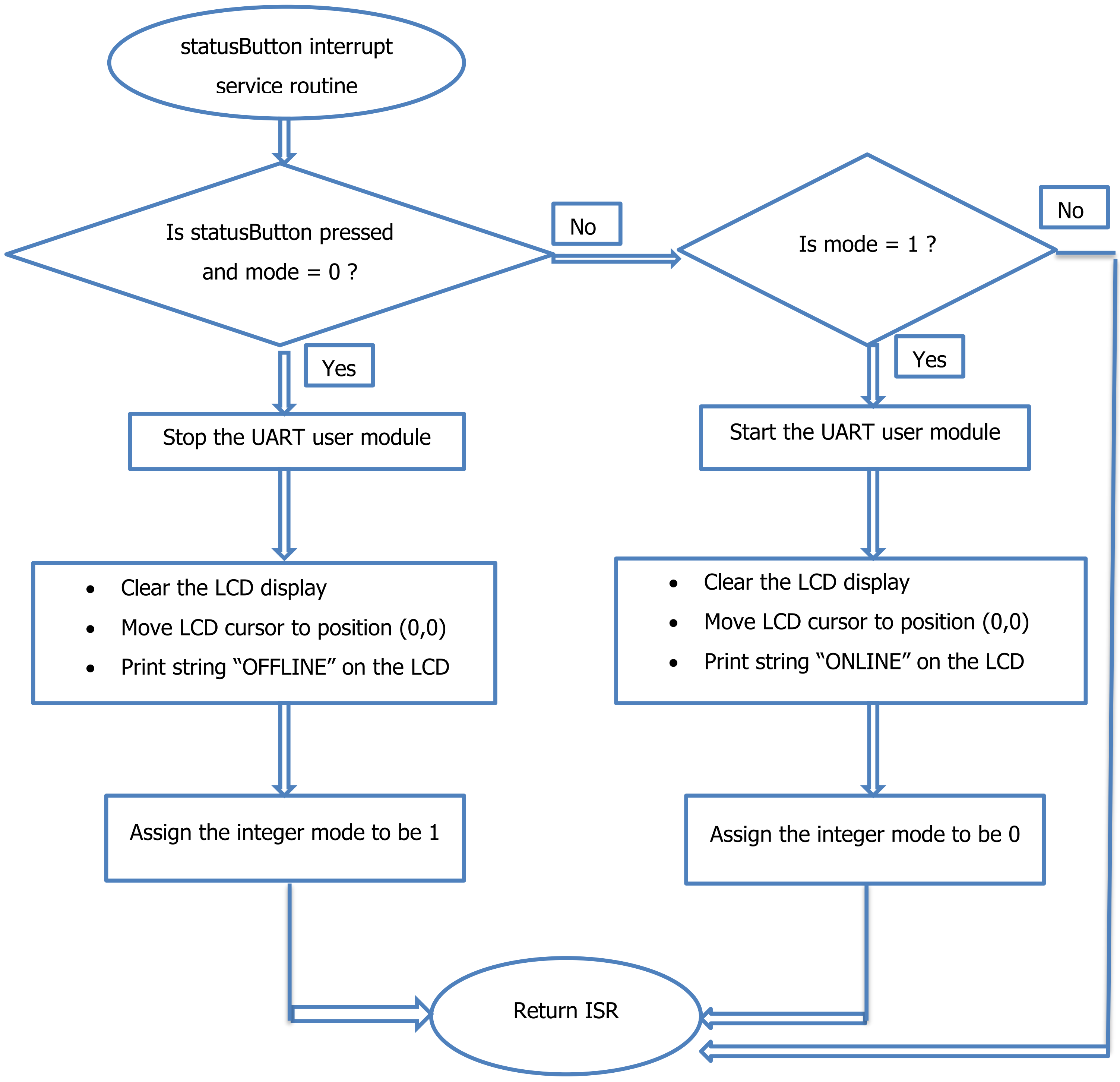

statusButton interrupt service routine
Is statusButton pressed and mode = 0 ?
No
Yes
Is mode = 1 ?
No
Yes
Stop the UART user module
Start the UART user module
Clear the LCD display
Move LCD cursor to position (0,0)
Print string "OFFLINE" on the LCD
Clear the LCD display
Move LCD cursor to position (0,0)
Print string "ONLINE" on the LCD
Assign the integer mode to be 1
Assign the integer mode to be 0
Return ISR

Figure 4. Flowchart for the interrupt service routine of the statusButton

Finally, the flowcharts for subroutines are given.

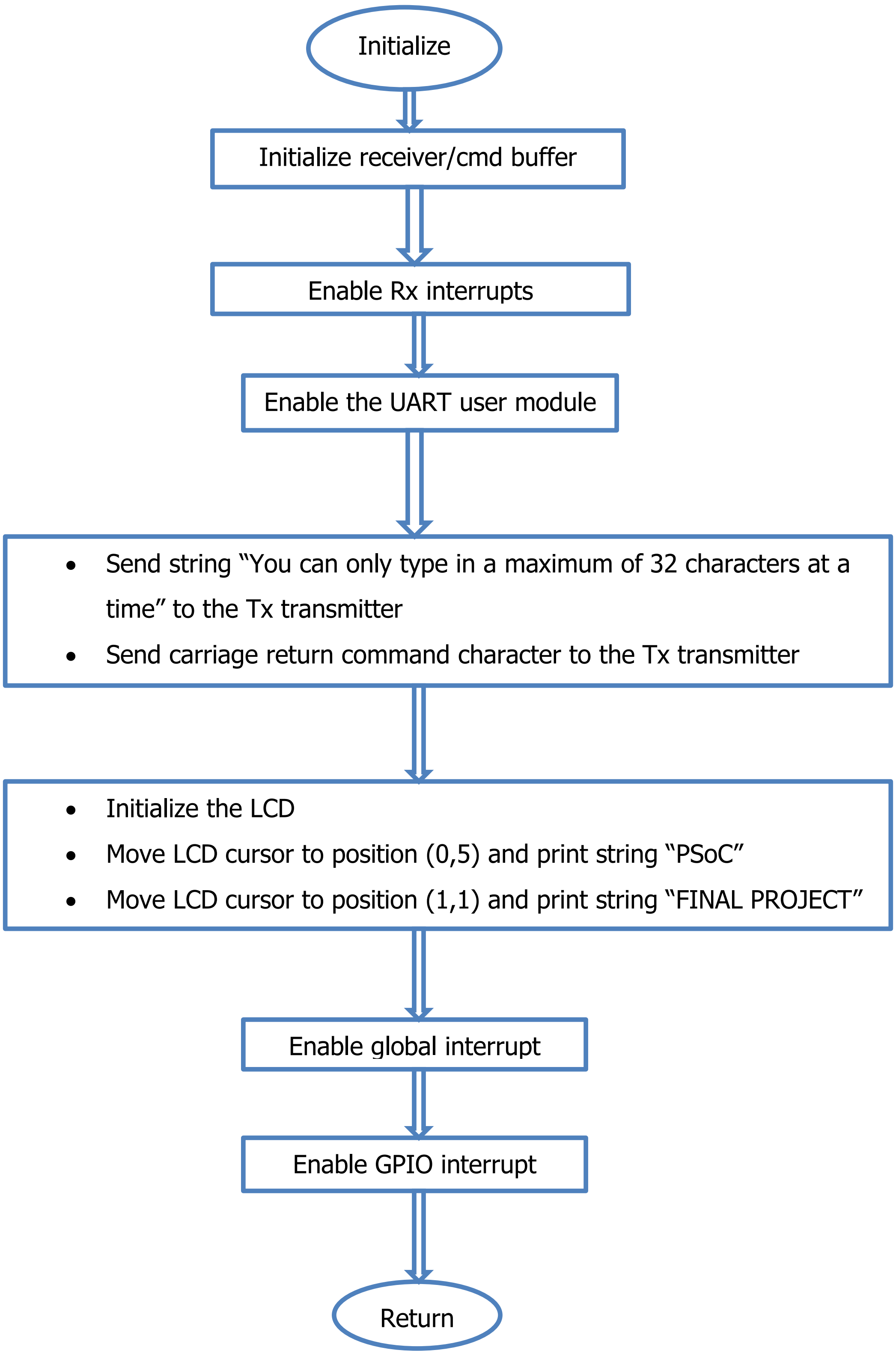

Initialize
Initialize receiver/cmd buffer
Enable Rx interrupts
Enable the UART user module
Send string "You can only type in a maximum of 32 characters at a time" to the Tx transmitter
Send carriage return command character to the Tx transmitter
Initialize the LCD
Move LCD cursor to position (0,5) and print string "PSoC"
Move LCD cursor to position (1,1) and print string "FINAL PROJECT"
Enable global interrupt
Enable GPIO interrupt
Return

Figure 5. Flowchart of subroutine for initializing the system

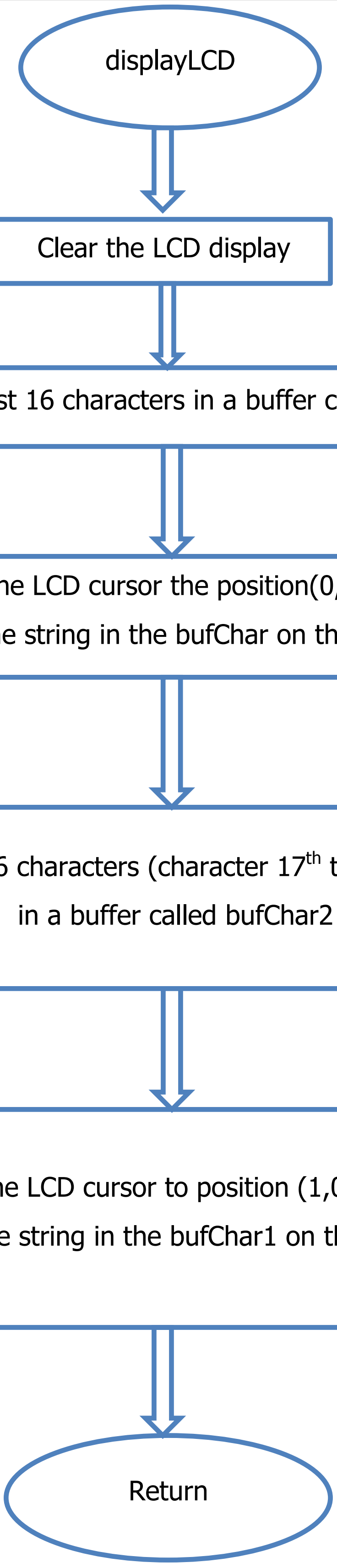

displayLCD
Clear the LCD display
Store the first 16 characters in a buffer called bufChar
Move the LCD cursor the position(0,0)
Print the string in the bufChar on the LCD display
Store the last 16 characters (character 17th to character 32nd) in a buffer called bufChar2
Move the LCD cursor to position (1,0)
Print the string in the bufChar1 on the LCD display
Return

Figure 6. Flowchart of subroutine for displaying data (text) on the LCD

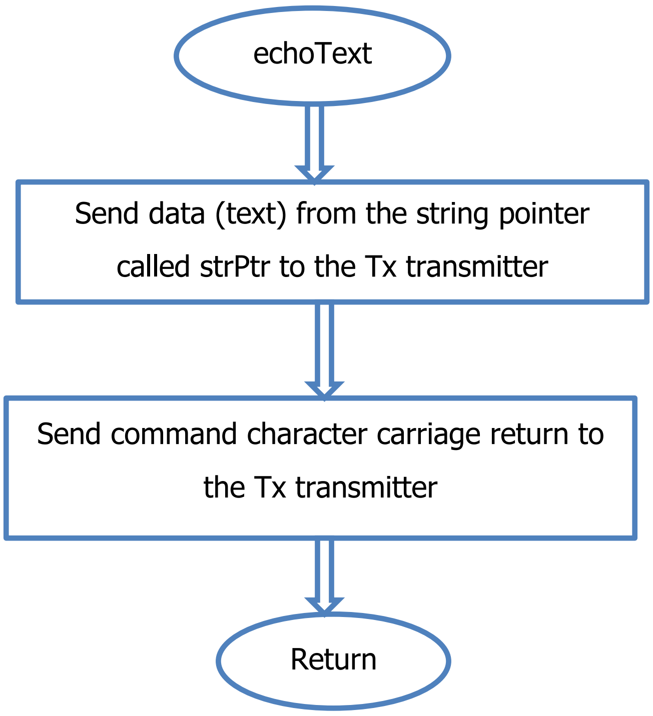

echoText
Send data (text) from the string pointer called strPtr to the Tx transmitter
Send command character carriage return to the Tx transmitter
Return

Figure 7. Flowchart of subroutine for sending data (text) back to the terminal

5 Testing the software

In the software design process, three subroutines were called. Those three subroutines fall though the structure to get full statement one by one. No conditions were used. Hence, there is no need to take these subroutines into account into the software testing phase.

In the main function and interrupt service routine, however, branch conditions were used in the algorithm. For that reason, it is necessary that software testing methods be applied to check if there are bug in these structures.

There are two methods which were used in the project for software testing process. They are functional testing (black-box testing) and coverage testing (white-box testing). First the main function was checked by using coverage testing method. Since the condition in the main function is atomic, there are two possibilities. The condition will check if a command terminator has been received. If the condition is "TRUE", then it the program will be executed in the next command line. IF the condition is "FALSE", then it will loop back to the forever loop.

With respect to the interrupt service routine for the statusButton, there are two conditions in the structure, and therefore, three test cases were applied to test the program. The following table will describe all the combinations of the three conditions.

Test case	Condition1	Condition2
#1	TRUE	DON'T CARE
#2	FALSE	FALSE
#3	FALSE	TRUE

Table 6. Test cases for the conditions

Condition1 checks if the statusButton has been pressed and if the integer mode is equal to 0. If the condition is "TRUE", then it will continue executing all the operations the top to the bottom. If the condition1 is "FALSE" (meaning that one of the two conditions is not satisfied with the condition1), then it will branch to condition2 in which the integer mode will be checked if mode is equal to 1. Now there are two possibilities. If the condition2 is "TRUE", then the operations following it will be executed. Otherwise, the condition is "FALSE", and the interrupt service routine will be return to the main function.

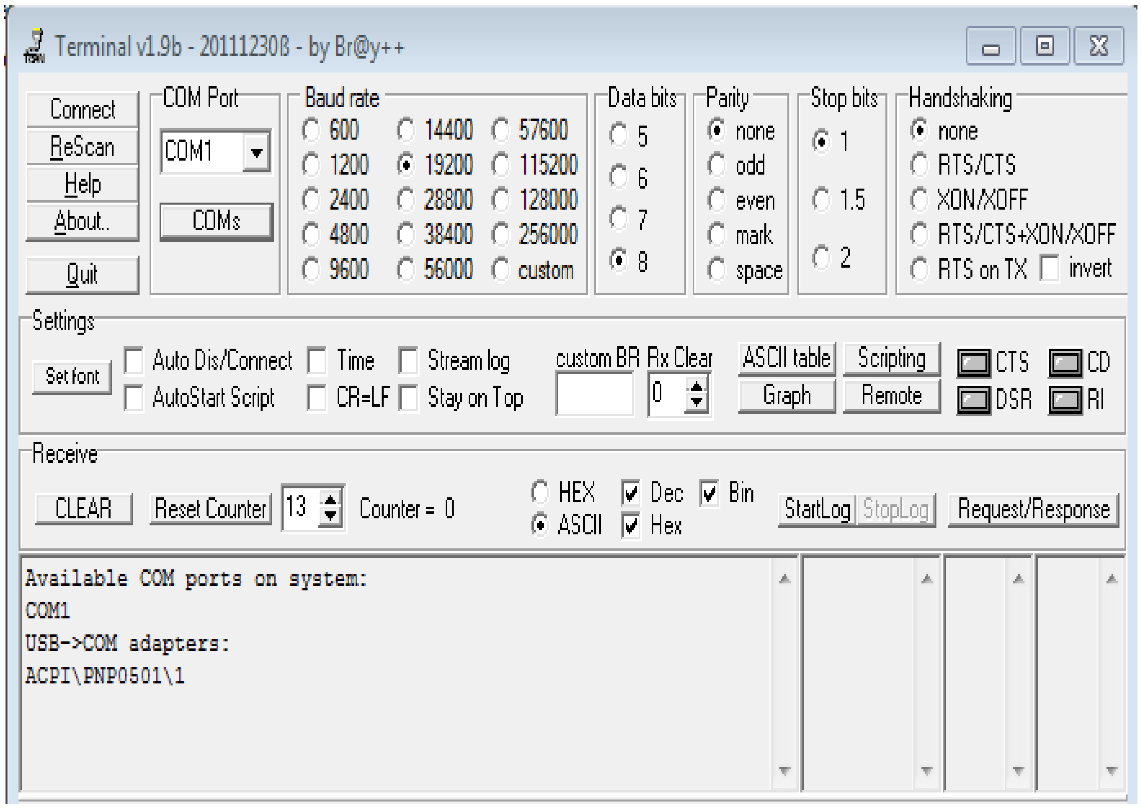
After the white-box testing, the black-box testing method was employed to check how well the implementation meets the requirements specification. The requirements specification is that after data (text) is typed from the terminal of the PC using a keyboard, the data (text) is displayed on the LCD and is also echoed back to the terminal. The second requirement is that when a button is pressed, the communication line will change from online mode to offline mode and vice versa. After the hardware and the terminal had been and configured properly (supposedly), the black-box testing was performed. A sample text "Hello, Finland!" was typed in the terminal. After that, the text "Hello, Finland!" was displayed on the LCD and echoed back to the terminal, confirming the success of the software as well as hardware. Having checked the first requirement, the requirement for the button was check. When then button was pressed, the string "OFFLINE" was displayed on the LCD. To check if it works, a sample text "Hello, Finland!" was typed in from the terminal and sent to the echo device. The result was nothing, which reflects that the offline mode works. When the button was pressed again, the string "OFFLINE" was displayed on the LCD. When the text "Hello, Finland!"

was typed in the terminal, the text was echoed back to the terminal and it was also displayed on the LCD. So the black-box testing has been successful.

6 Operation of the echo device

All hardware settings from the device configuration are loaded into the device and main.c is executed on program execution. The 24-MHz system clock is divided by 156 (VC3) to produce a 153.8-KHz clock, which is provided to the UART user module. The transfer rate is 1/8 times the clock, that is, 19.2 kbps. Parity is set to none in the initial configuration for the block.

The settings for the Terminal are as follows:



COM Port: COM1

Baud rate: 600 1200 2400 4800 9600 14400 [✓ 19200] 28800 38400 56000 57600 115200 128000 256000 custom

Data bits: 5 6 7 [✓ 8]

Parity: [✓ none] odd even mark space

Stop bits: [✓ 1] 1.5 2

Handshaking: [✓ none] RTS/CTS XON/XOFF RTS/CTS+XON/XOFF RTS on TX ☐ invert

Receive: 13 [✓ ASCII] HEX ✓ Dec ✓ Hex ✓ Bin

```
Available COM ports on system:
COM1
USB->COM adapters:
ACPI\PNP0501\1
```

Figure 8. Settings for the terminal on the PC

In the COM Port → choose COM1

Baud rate → choose 19200

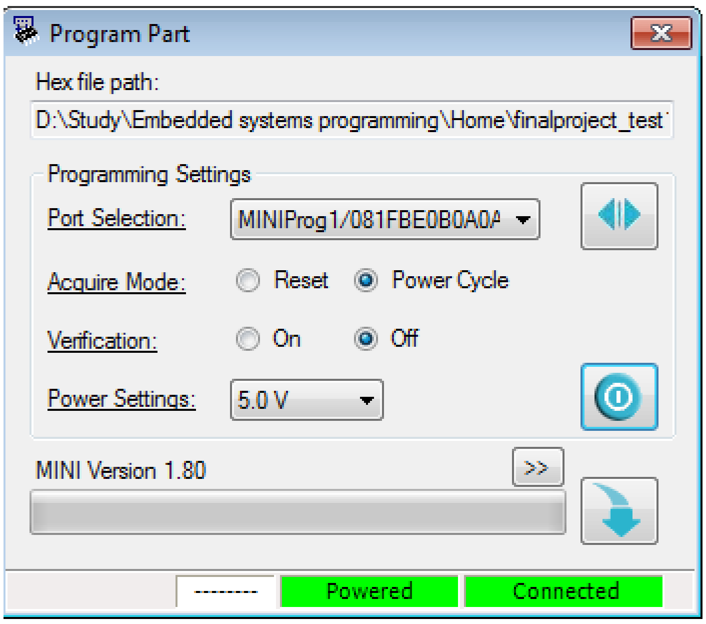
Data bits → choose 8

Parity → none

Stop bit → 1

Handshaking → none

After setting the terminal, power up the PSoC by choosing the menu Program from the PSoC Designer, and then choose Program Part (or ctrl + F10). A pop-up menu appears on the screen → click on the toggle power.



Program Part

Hex file path:
D:\Study\Embedded systems programming\Home\finalproject_test

Programming Settings

Port Selection: MINIProg1/081FBE0B0A0A

Acquire Mode: Reset [✓ Power Cycle]

Verification: On [✓ Off]

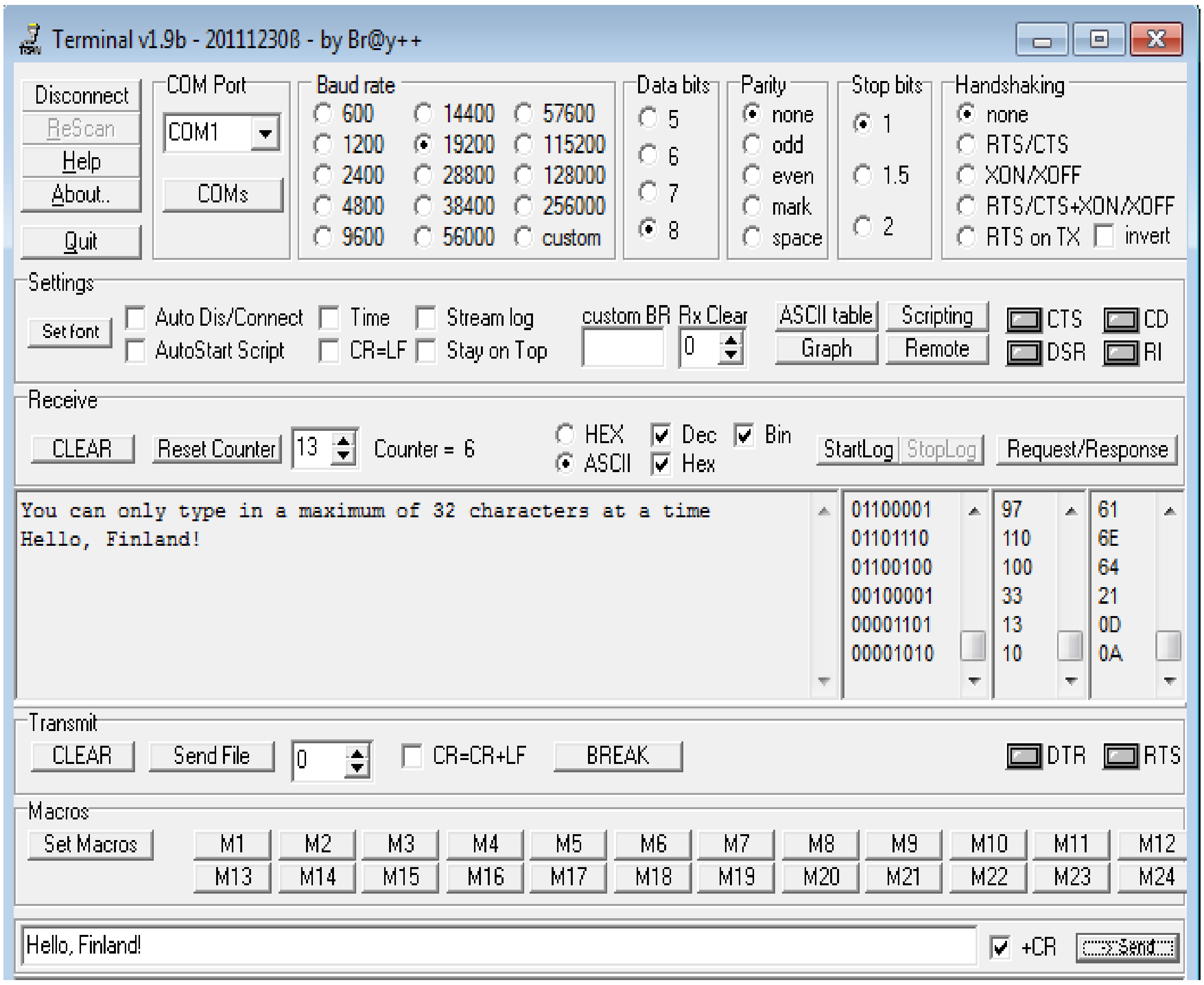
Power Settings: 5.0 V

MINI Version 1.80

-------- Powered Connected

Figure 9. Powering up the PSoC platform

After powering up the PSoC platform, the string "You can only type in a maximum of 32 characters at a time" is displayed on the terminal of the PC. To demonstrate the data (text) inputted from the terminal will be sent to the LCD display and then echoed back to the terminal, a sample string "Hello, Finland!" is typed in from the terminal. The result can be seen from the following figures.



COM Port: COM1
Baud rate: 19200
Data bits: 8
Parity: none
Stop bits: 1
Handshaking: none

Receive: [✓ ASCII] [✓ Dec] [✓ Bin] [✓ Hex] 13 Counter = 6

```
You can only type in a maximum of 32 characters at a time
Hello, Finland!
```

```
01100001   97   61
01101110   110  6E
01100100   100  64
00100001   33   21
00001101   13   0D
00001010   10   0A
```

Transmit: 0

Hello, Finland! [✓ +CR]

Figure 10. The text is inputted from the terminal and echoed back to the terminal

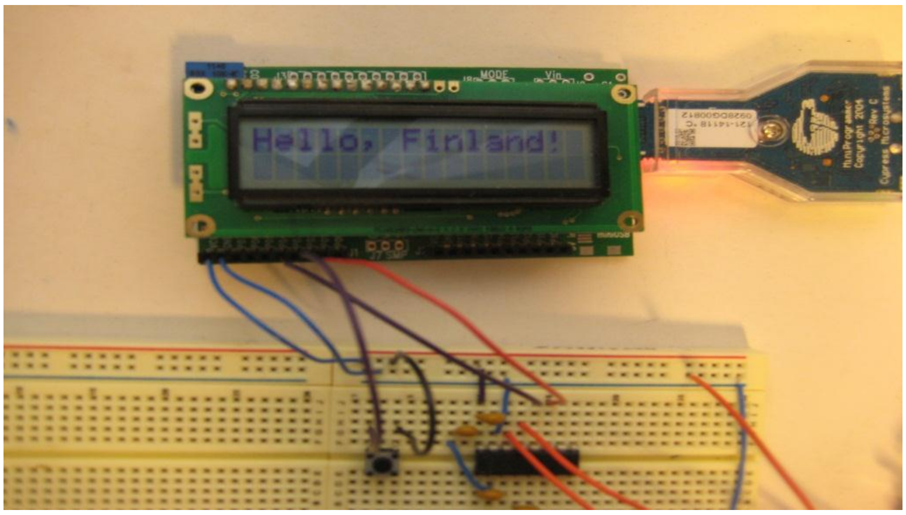
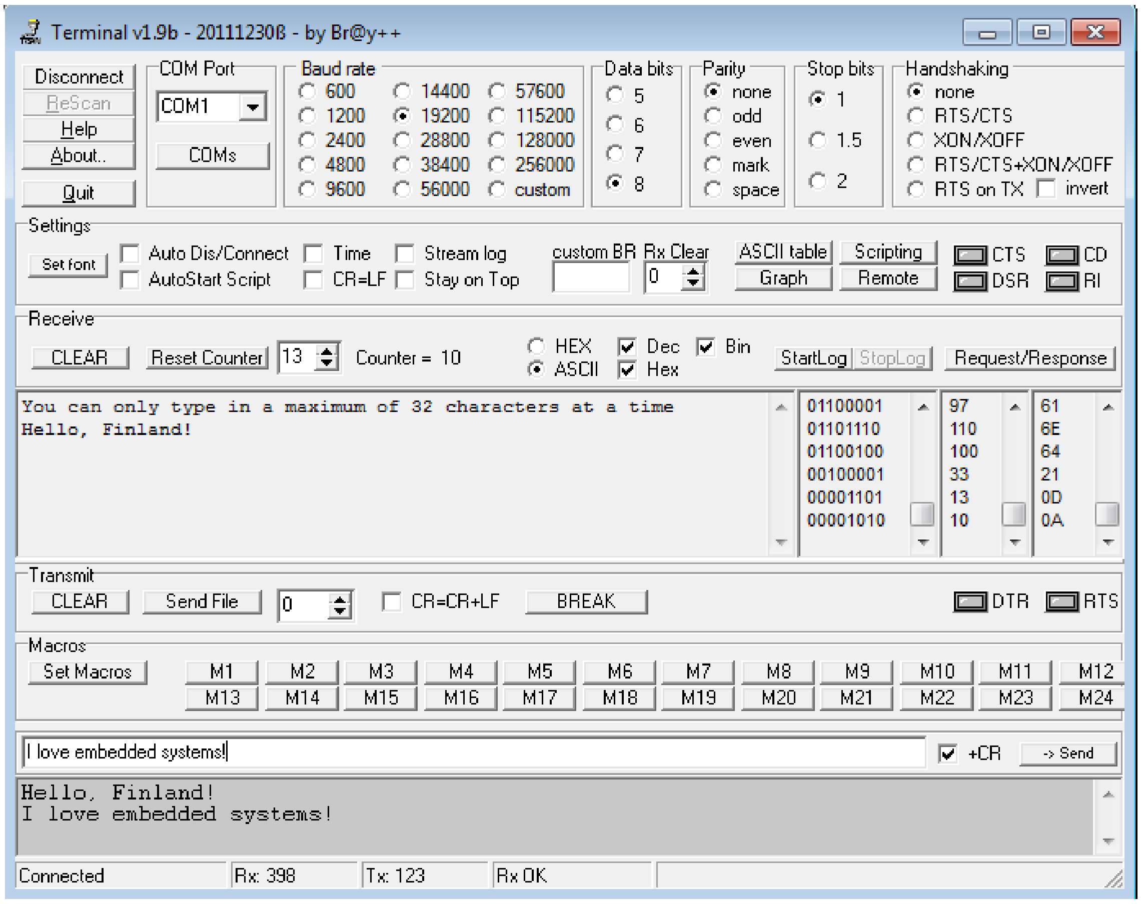


Figure 11. The text is inputted from the terminal and displayed on the LCD display

Now a demonstration for the online/offline mode when the pushbutton is pressed is shown. After the pushbutton has been pressed, the string "OFFLINE" is shown on the

LCD display, indicating that the serial communication line is now in offline mode, and no data can be displayed when this offline mode is applied. A sample text "I love embedded systems!" is typed in the terminal and sent out as can be seen in figure 12.



Terminal v1.9b - 20111230ß - by Br@y++

COM Port: COM1 · Baud rate: 19200 · Data bits: 8 · Parity: none · Stop bits: 1 · Handshaking: none

Receive: 13 · Counter = 10 · ASCII · Dec ✓ · Hex ✓ · Bin ✓

```
You can only type in a maximum of 32 characters at a time
Hello, Finland!
```

```
01100001   97    61
01101110   110   6E
01100100   100   64
00100001   33    21
00001101   13    0D
00001010   10    0A
```

Transmit: 0

I love embedded systems! · +CR ✓

```
Hello, Finland!
I love embedded systems!
```

Connected | Rx: 398 | Tx: 123 | Rx OK

Figure 12. Another sample text is inputted in offline mode of the echo device

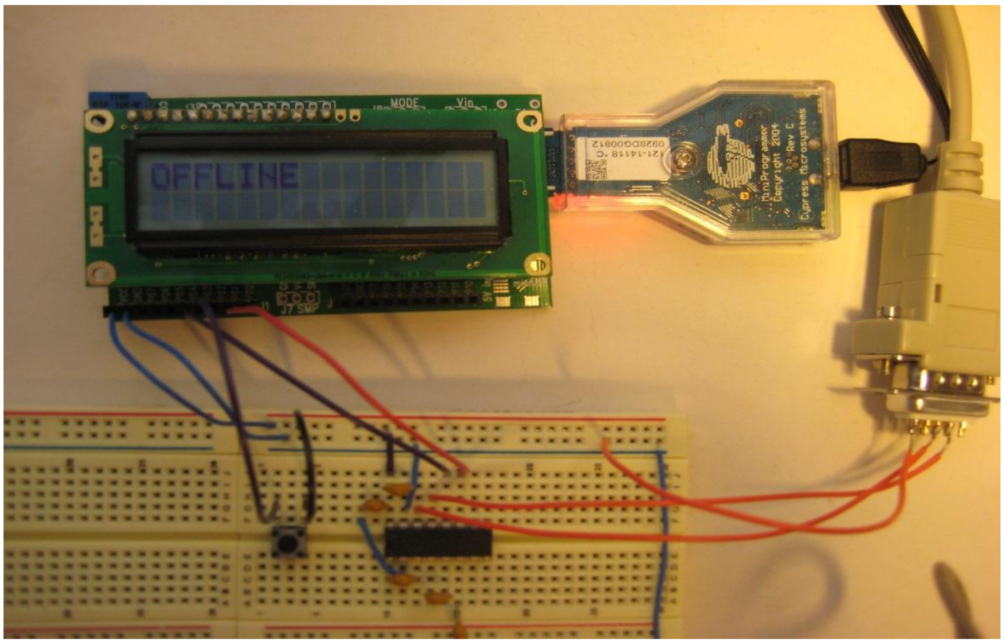


Figure 13. The echo device in offline mode

Looking at figure 13, it can be seen that the text "I love embedded systems!" was sent to the echo device in its offline mode, even though text had been sent out from the terminal. In addition, the text was not echoed back to the terminal as can be observed in figure 12.

Now the pushbutton is pressed again, and it is now in online mode.

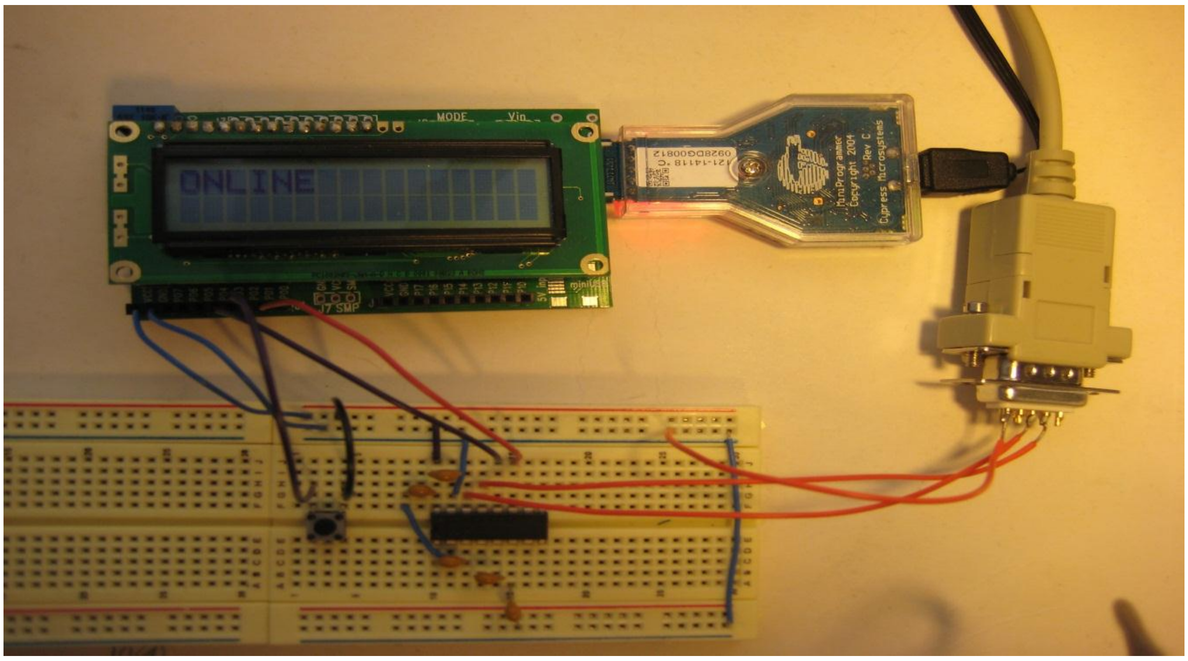
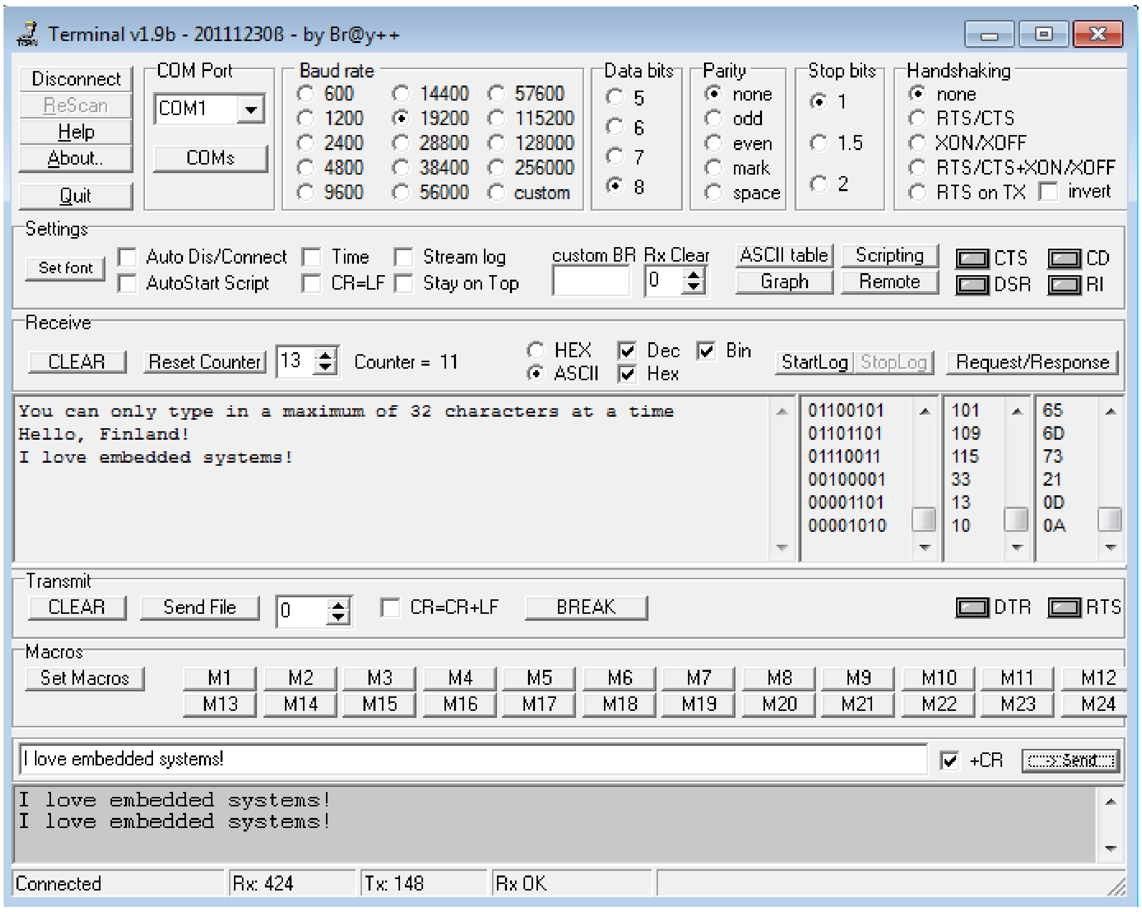


Figure 14. The echo device in online mode after pressing the pushbutton

After the echo device has been in online mode, the text “I love embedded systems!” is inputted again to the terminal and sent over the serial communication line. The text “I love embedded systems!” is then displayed on the LCD and is also echoed back to the terminal. The following figures 15 and 16 illustrate the results.



Terminal v1.9b - 20111230B - by Br@y++

COM Port: COM1
Baud rate: [✓ 19200]
Data bits: [✓ 8]
Parity: [✓ none]
Stop bits: [✓ 1]
Handshaking: [✓ none]

Receive: Reset Counter 13, Counter = 11; [✓ ASCII], [✓ Dec], [✓ Bin], [✓ Hex]

```
You can only type in a maximum of 32 characters at a time
Hello, Finland!
I love embedded systems!
```

```
01100101   101   65
01101101   109   6D
01110011   115   73
00100001   33    21
00001101   13    0D
00001010   10    0A
```

Transmit:

```
I love embedded systems!
```

[✓ +CR]

```
I love embedded systems!
I love embedded systems!
```

```
Connected    Rx: 424    Tx: 148    Rx OK
```

Figure 15. The text “I love embedded systems!” is sent and echoed back

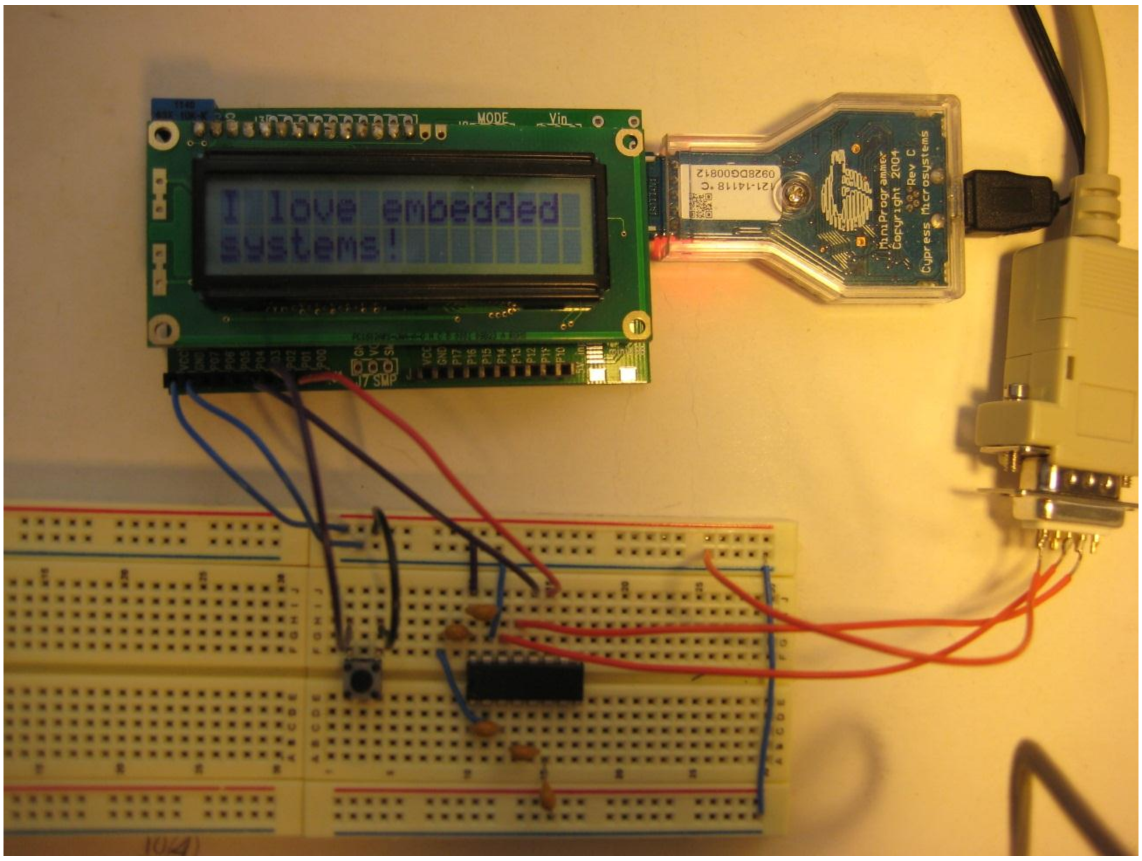


Figure 16. The text "I love embedded systems!" is displayed on the LCD in online mode

Now is the end of the operation of the echo device. For more detailed information about the operation of the echo device, please follow the link below.
http://www.youtube.com/watch?v=idyJYWaXOG0
Note: The quality of the video in youtube should be switch to HD for better quality.

7 Conclusions and recommendations

To put it in a nutshell, the project is about devising an echo device which is able to communicate with the PC via serial communication. Also, the mode of the communication (online/offline) can be selected by the user when a pushbutton is pressed.

From the project, I have learnt one more PSoC block, the UART, and about the circuit for getting 3.3 to 5V TTL device to talk to the PC with an RS-232 serial port using MAX232ACPE.

After finishing the echo device project, I have gained some experience and now I will give some recommendations based on the experience. First, connection from the pins of DB-9 to the pins of MAX232ACPE should be taken into account, especially when reading the datasheet. Checking if the serial connector is female or male is important, so information from the datasheet should be scrutinized. I made a serious mistake in connecting the pins of DB-9 cable to the pins of the MAX232ACPE. Consequently, it took me 2 days to figure out whether it was wrong with the hardware or the software. After that, the problem was fixed by using the sample firmware from the datasheet of the UART so that I can make absolute sure there is no bug with the software. Running the PSoC platform with the sample code, the problem was not solved. And I knew that there was, for sure, a problem with the hardware. The problem was limited to only the hardware issue, and it was then solved quickly by connecting the pins of DB-9 connector to the pins of the MAX232ACPE again.

Regarding the limitation of the project, the LCD display in use for the project is a 16 × 2 LCD, so a maximum of 32 characters can be displayed at a time. If the requirements specification for the project is more than 32 characters at a time, then another LCD display should be employed to meet the requirement.

References

1. Jean J. Labrosse. Embedded Systems Building Blocks, 2nd edition, R&D Books, the U.S, p. 400-416.
2. RS-232, accessed 18.05.2012, http://en.wikipedia.org/wiki/RS-232