Chapter 11. A Voice-Synthesizer Project

This chapter is the extension of the previous chapter so that we generate voice from the typed words from the keyboard, using a voice synthesizer board. Imagine that a person communicate in a written form and wants it to be spoken. So the person can see what is s typed and the party can hear what the person intends to say. This feature needs a voice synthesizer which does the text-to-voice conversion.

1. DoubleTalk RC8650 Voice Synthesizer

The DoubleTalk RC8650 is versatile voice and sound synthesizers, integrating a sophisticated text-to-speech processor, audio recording and playback, musical and sinusoidal tone generators, telephone dialer and A/D converter, all in easy to use chipsets. This chipset translate plain English text into speech in real time, without the assistance of a PC or high-powered processor. It enables us to add text-to-speech capability to virtually any design, quickly and painlessly.

In addition, integrated tone generators provide telephone dialing, music, and programmable signaling tones. Up to 3.5 MB of built in, flash-based recording memory can store up to 15 minutes of sound files, which can be played back on demand by the host.

The RC8650 chip set is comprised of two surface-mounted devices: the RC8650 and RC4651. Both operate from a +5 V supply and consume very little power. In many cases, all that is needed to build a fully functional system is a low pass filter and audio amplifier (which can often be combined into the same circuit).

As text messages are sent to the RC8650, the RC8650 automatically converts the messages into speech using an integrated text-to-speech processor. The TTS processor utilizes RC Systems' DoubleTalk TTS technology, which is based on a patented voice concatenation technique using real human voice samples. Voice control parameters, such as speed, volume, tone, pitch and expression, can also be embedded within the text stream for dynamic on-the-fly voice control. RS-232 compatible serial and 8 bit bus interfaces are included to allow the chipset to interface to virtually any CPU or microcontroller.

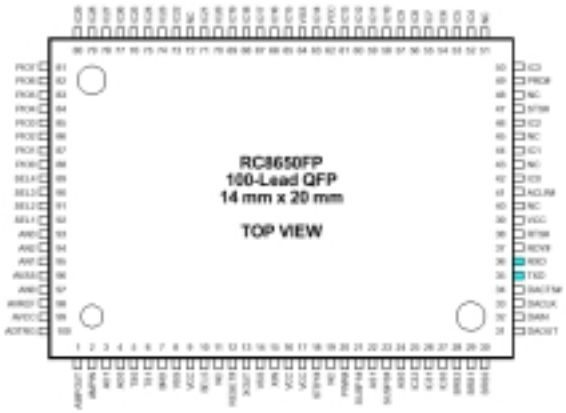

Fig. 79 RC8650FP

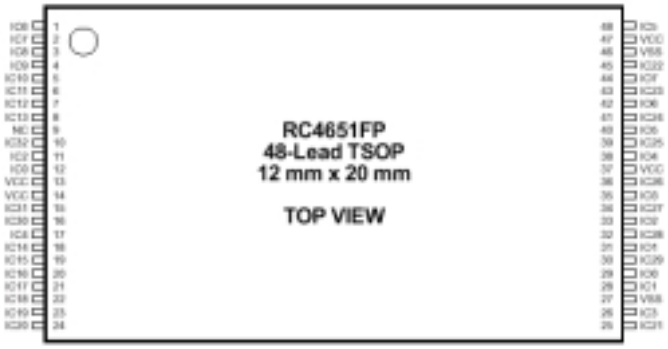

Fig. 80 RC4651FP

The DoubleTalk RC8650 Evaluation Kit enables you to experiment with the RC Systems RC8650 voice synthesizer chip set. Included in the kit are:

- Evaluation board containing the RC8650 chip set
- Speaker with volume control
- Serial cable
- RC8650 Studio software

The evaluation board is a complete, versatile voice synthesizer which can be used with the RC8650 Studio software as well as in stand-alone applications. The board includes the RC8650 voice synthesizer chip set, audio power amplifier, voltage regulator, RS-232C interface, and parallel I/O port. The chip set's I/O lines are made accessible through header connectors near the edge of the board. SW1 in the evaluation board is the Reset switch. Press once when we meet some problem.

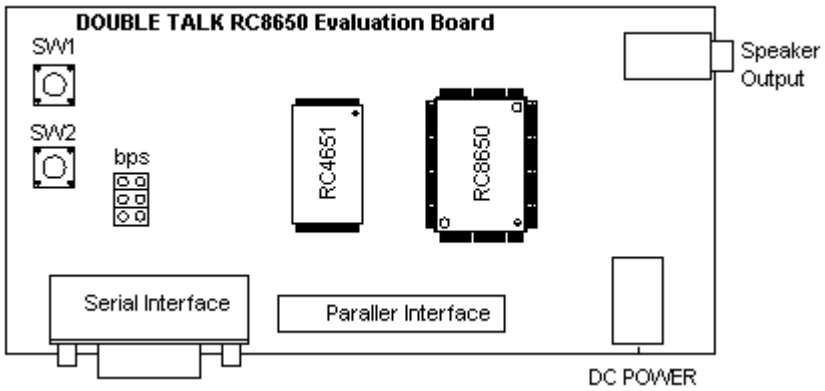

Fig. 81 RC8650 Evaluation board

The RC8650 Studio software is NOT required, however, in order to use the evaluation board. The board can be used "stand-alone" if desired by simply printing the desired text and commands to it via the board's serial or parallel ports.

So we directly tap the RXD and TXD pin of RC8650 chip set for direct serial communication with 16F877 without passing through RS-232 level converter such as MAX232. In this stand-alone application, a text word typed is voiced after a CR key is provided to RC8650 voice synthesizer.

2. Operating Modes of RC8650 Chip Set

The RC8650 has six primary operating modes and two low-power modes designed to achieve maximum functionality and flexibility. The operating mode can be changed anytime, even on the fly. Note The RC8650 does not make any distinction between uppercase and lowercase characters. Text and commands may be sent as all uppercase, all lowercase, or any combination thereof.

Text mode. In this mode, all text sent to the RC8650 is spoken as complete sentences. Punctuation is also taken into consideration by the intonation generation algorithms. The RC8650 will not begin speaking until it receives a CR (ASCII 13) or Null (ASCII 00) character—this ensures that sentence boundaries receive the proper inflection. This is the default operating mode.

Character mode. This mode causes the RC8650 to translate input text on a character-by-character basis; i.e., text will be spelled instead of spoken as words. The RC8650 does not wait for a CR/Null in this mode.

Phoneme mode. This mode disables the RC8650's text-to-phonetics translator, allowing the RC8650's phonemes to be directly accessed. Phonemes in the input buffer will not be spoken until a CR or Null is received.

Real Time Audio Playback mode. In this mode, data sent to the RC8650 is written directly to its audio buffer. This results in a high data rate, but provides the capability of producing the highest quality speech, as well as sound effects. PCM and ADPCM data types are supported.

Prerecorded Audio Playback mode. This mode allows recorded speech and sound effects to be stored on-chip and played back at a later time. PCM and ADPCM data types are supported.

Tone Generator modes. These modes activate the RC8650's musical tone generator, sinusoidal generator, or DTMF generator. They can be used to generate audible prompts, music, signaling tones, dial a telephone, etc.

Idle mode. To help conserve power in battery-powered systems, the RC8650 automatically enters a reduced-power state whenever it is inactive. Data can still be read and written to the RC8650 while in this mode. Current draw in this mode is typically 1 mA.

Standby mode. This mode powers down the RC8650, where current draw is typically only 2 µA. Standby mode can be invoked from either the STBY# pin or with the Sleep command. Data cannot be read from or written to the RC8650 in this mode.

3. Commands of RC8650

The commands described in the following pages provide a simple yet flexible means of controlling the RC8650 under software control. They can be used to vary voice attributes, such as the volume or pitch, to suit the requirements of a particular application or listener's preferences. Commands are also used to change operating modes. Commands can be freely intermixed with the text that is to be spoken, allowing the voice to be dynamically controlled. Commands affect only the data that follows them in the data stream.

The command character. The default RC8650 command character is Control-A (ASCII 01). The command character itself can be spoken by the RC8650 by sending it twice in a row: Control-A Control-A. This special command allows the command character to be spoken without affecting the operation of the RC8650, and without having to change to another command character and then back again.

Command Syntax. All RC8650 commands are composed of the command character, a parameter n comprised of a one to three-digit number string, and a single string literal that uniquely identifies the command. Some commands simply enable or disable a feature of the RC8650 and do not require a parameter. The general command format is:
<command character>[<number string>]<string literal>
If two or more commands are to be used together, each must be prefaced with the command character. This is the only way the RC8650 knows to treat the remaining characters as a command, rather than text that should be spoken. For example, the following commands program pitch level 40 and volume level 7 (Control-A is the default command character):

```
Control-A "40P" Control-A "7V"
```

4. Some Global Commands of RC8650

Voice (nO). The text-to-speech synthesizer has eight standard voices and a number of individual voice controls that can be used to independently vary the voice characteristics. Voices are selected with the commands 0O through 7O, shown in Table 2.3. Because this command alters numerous internal voice parameters (pitch, expression, tone, etc.), it should precede any individual voice control commands.

n	Voice Name
0	Perfect Paul(default)
1	Vader
2	Big Bob
3	Precise Pete
4	Ricochet Randy
5	Biff
6	Skip
7	Robo Robert

Volume (nV). This is a global command which controls the RC8650's output volume level, from 0V through 9V. 0V yields the lowest possible volume; maximum volume is attained at 9V. The default volume is 5V. The Volume command can be used to set a new listening level, create emphasis in speech, or change the output level of the tone generators.

DTMF Generator (n*). The DTMF (Touch-Tone) generator generates the 16 standard tone pairs commonly used in telephone systems. Each tone pair generated by the RC8650 is 100 ms in duration, more than satisfying the telephone signaling requirements (this can be extended to 500 ms with the Protocol Options Register command). The mapping of the command parameter n to the buttons on a telephone is shown below. The "pause" tone is used to generate the inter-digit delay in phone number strings. The generator's output level can be adjusted with the Volume command (nV). DTMF commands are buffered, and may be intermixed with text and other commands without restriction.

n	Button
0	0
-	-
-	-
9	9
10	*
11	#
12	A
13	B
14	C
15	D
16	pause

5. Coding Example for RC8650

As mentioned above we tap the RXD (pin#35) and TXD (pin# 36) of RC8650 chip for serial communication with 16F877. Since we need the hardware implemented serial communication and the MAX232 for hex code download from the PC we work for coding, we utilize the software implemented serial communication (refer to Chapter 6) for the connection with RC8650, and we pick RD5 and RD4 for RX and TX pins for 16F877, respectively.

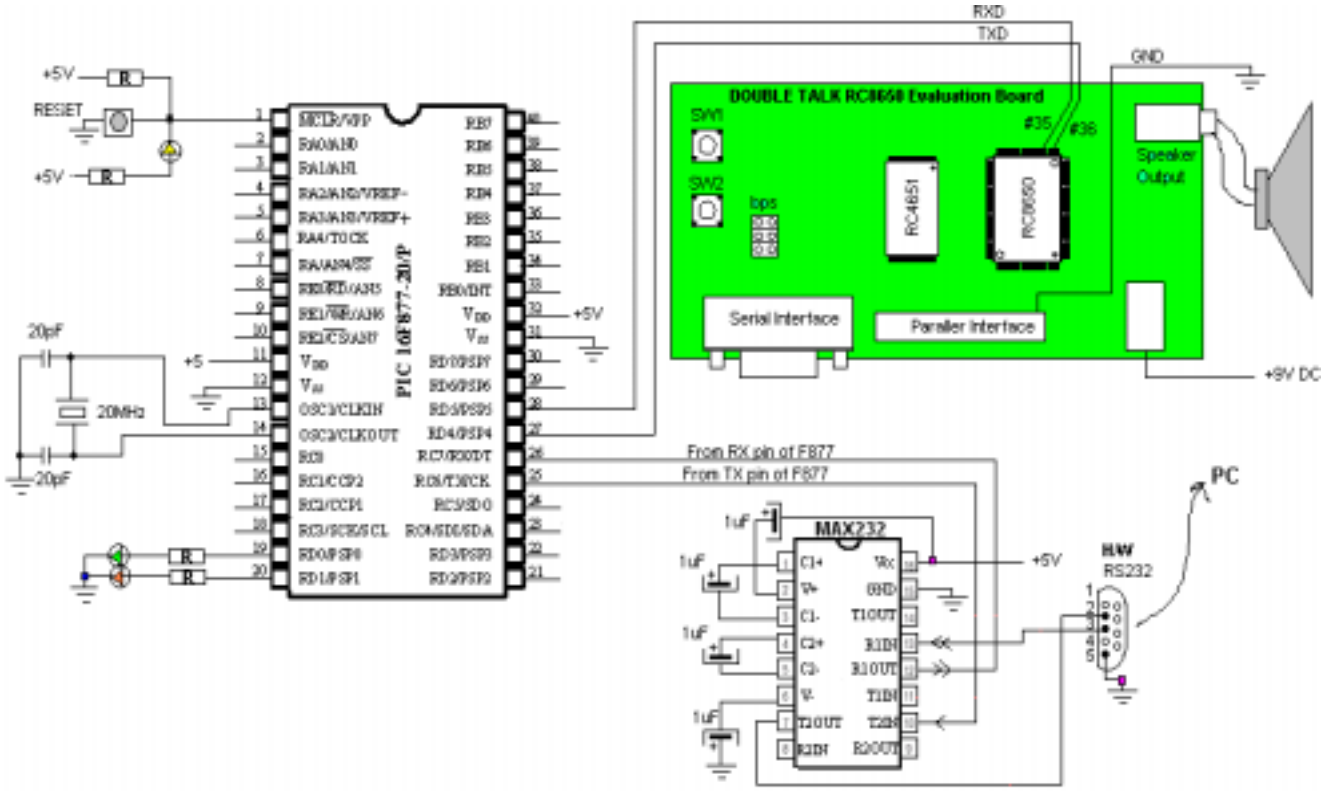

Fig. 82 RC8650 connection to PIC 16F877

The example code is to simply dial a number and generate the voice for a text of "I LOVE YOU." We will test all 8 different voices supported by the chip set and control volume of the voice.

First we can set the volume to 4 by the following routine. Since every command should start with the command character Control-A, we start with it and use the global command format for volume. Note that every command and attribute is entered as character. `CTRLA` is declared as 0x01 ASCII code. Subroutine `TXSW` is 19200 bps software generated serial communication (transmission) routine that we already discussed in Chapter 6.

```
;
      movlw       CTRLA
      call        TXSW
      movlw       '4'
      call        TXSW
      movlw       'V'
      call        TXSW        ;Send CTRL-A > 4 > V
```

For voice, we want to apply all 8 different voices, so we first assign 0 to a register, `VOICE`, and increase by 1 after we generate the sound of the text "I Love You". By the way, the default operational mode of RC8650 chip set is the text-mode which coverts text to voice.

The phone dialing precedes the "I Love You" message. The phone dialing, or DTMF generation, is done by the following order of command and each digit of phone number:

CTRL-A > Digit_1>*>CTRL-A>Digit_2>* .....CTRL-A>Digit_n>*.
In other words, the CTRL-A should come before the number and it should be followed by the star (*) mark.

The following instructions dial and generate DTMF of the author's office number:202-806-4821 with preceding '1' for long distance call indication.

```
movlw       CTRLA
call        TXSW
movlw       '1'
call        TXSW
movlw       '*'
call        TXSW


movlw       CTRLA
call        TXSW
movlw       '2'
call        TXSW
movlw       '*'
call        TXSW


movlw       CTRLA
call        TXSW
movlw       '0'
call        TXSW
movlw       '*'
call        TXSW


movlw       CTRLA
call        TXSW
movlw       '2'
call        TXSW
movlw       TXSW


movlw       CTRLA
call        TXSW
movlw       '8'
call        TXSW
movlw       '*'
call        TXSW


movlw       CTRLA
call        TXSW
movlw       '0'
call        TXSW
movlw       '*'
call        TXSW


movlw       CTRLA
```

```
      call        TXSW
      movlw       '6'
      call        TXSW
      movlw       '*'
      call        TXSW


      movlw       CTRLA
      call        TXSW
      movlw       '4'
      call        TXSW
      movlw       '*'
      call        TXSW


      movlw       CTRLA
      call        TXSW
      movlw       '8'
      call        TXSW
      movlw       '*'
      call        TXSW


      movlw       CTRLA
      call        TXSW
      movlw       '2'
      call        TXSW
      movlw       '*'
      call        TXSW


      movlw       CTRLA
      call        TXSW
      movlw       '1'
      call        TXSW
      movlw       '*'
      call        TXSW
```

This long lines of the code can be simplified by developing a subroutine and a table. First, let's make a table for the phone number:

```
;=== Phone Number Table=
PhoneTable
      movf        PHONEdigit,0
      addwf       PCL
                                ;PC+0
      DT          "12028064821"     ;11 numbers
      retlw       0
```

As we see above it looks very clean and simple with table format. Now let's have a dialing subroutine which dials the numbers stored in the `PhoneTable` table. In the `Dialing` subroutine, the PhoneTable is called 11 times, and at each time, with PC increased, the next number is restored to W register for writing to RC8650. The command CTRL-A and star mark (*) are wrapping the phone number for DTMF generation.

```
Dialing     movlw       0x0B        ;11 phone digits
            movwf       Ptemp
            clrf        PHONEdigit
Dagain      movlw       CTRLA
            call        TXSW
            call        PhoneTable        ;get the number
            call        TXSW
            movlw       '*'
            call        TXSW
            incf        PHONEdigit
            decfsz      Ptemp
            goto        Dagain
            return
```

The burst duration of the DTMF is 100ms in the default setting of the chip set. We can change the duration to 500ms by changing the content of Protocol Options Register of the chip set. Details on this subject is left to the readers, and here goes the command for 500ms burst duration:
CTRL-A >"1">"6">"0">"G".

Selection of a voice comes with CTRL-A followed by the voice number, 0 through 7, and the letter O, as shown below

```
      movlw       CTRLA
      call        TXSW
      movf        VOICE,0           ;write the VOICE
      call        TXSW
      movlw       'O'
      call        TXSW
```

Since the text mode is the default mode of RC8650, write a message is pretty simple. For "I Love You! " message, we go like this:

```
      movlw       'I'
      call        TXSW
      movlw       ' '         ;space
      call        TXSW
      movlw       'L'
      call        TXSW
      movlw       'o'
      call        TXSW
      movlw       'v'
      call        TXSW
      movlw       'e'
      call        TXSW
      movlw       ' '   ;space
      call        TXSW
      movlw       'Y'
      call        TXSW
      movlw       'o'
      call        TXSW
      movlw       'u'
```

```
      call        TXSW
      movlw       'I'
      call        TXSW
```

As we did in phone number, the text message can also be simplified by a subroutine and a table. First, the message table looks like this:

```
MessageTable
      movf        MESSAGEdigit,0
      addwf       PCL
      DT          "I LOVE YOU!"     ;11 texts
      retlw       0
```

And the subroutine for text message writing goes as shown below. There are 11 character readings and writings without any other commands and command characters. When all characters in the text message are read, then CR is written to RC8650 to signal the end of the text message and to request for conversion to voice.

```
;Subroutine Message
Message
      movlw       0x0B              ;11 characters
      movwf       Mtemp
      clrf        MESSAGEdigit
Magain
      call        MessageTable      ;read a text
      call        TXSW
      incf        MESSAGEdigit
      decfsz      Mtemp
      goto        Magain
      movlw       0x0D
      call        TXSW        ;CR key for voicing after 11 readings
      return
```

Since this is the first incidence of RC8650 voice synthesizer application, the following code lists the full program. When you run this, you would hear very quick digital dialing sound from the speaker (100ms per each digit) followed by a voice saying "I Love You." This dialing and message repeats for 8 different voices.

```
;RC8650.asm
;
;TABLE IS USED TO SIMPLIFY THE PROGRAM
; FOR PHONE NUMBER
;AND
;TEXT MESSAGE
;
;This program is to:
; 1. test the RCS8650 voice synthesizer evaluation board
; 2. Send ASCII word followed by CR key
; 3. Then the sound must be generated from the speaker attached to the board
; 4. Connection
;  DB9 of 16F877 to DB9 of RC6850 Board
;
;  This connection is made without using MAX232 chips at both sides
```

```
;Direct connection between 16F877 and RC8650

;   Baud rate for this is set as 19200
;
;This program is asynchronous communication using software method
;
;F = 20 MHz
;B = Baud Rate
;For B=19200, one Baud cycle (BC) is about 52uS
;
;
;TRANSMIT MODE
;First START bit is sent by setting the TX pin to LOW for (BC) seconds
;And, from then on, the TX Pin is Set/Cleared corresponding to the data bit
;every (BC) seconds.
;8N1 format
;
;TX Pin = RD4
;RX Pin = RD5
;
;Terminal set up: 8N1 19200
;
;

        list P = 16F877

PCL         EQU   0x02
STATUS      EQU   0x03
CARRY       EQU   0x00
TRISD       EQU   0x88
PORTD       EQU   0x08
TXPIN       EQU   0x04        ;RD4
RXPIN       EQU   0x05        ;RD5
MSB         EQU   0x07
CTRLA       EQU   0x01        ;RC8650 Command Character
;
;note

;RAM for DELAY SUBROUTINE
        CBLOCK    0x20              ; RAM AREA for USE at address 20h
            PHONEdigit
            Ptemp
            MESSAGEdigit
            Mtemp
            VOICE
            Kount52us
            Kount100us
            Kount10ms
            Kount100ms
            Kount1s
            RCSreg            ;data to RCS's RC8650
            Bitcount    ;data bit count
            Kount       ;Delay count (number of instr cycles for delay)
      ENDC


;=======================================================
```

```
      org         0x0000            ;line 1
      GOTO        START       ;line 2 ($0000)
;=====================================================
      org         0x05

START

      banksel     TRISD
; Port setting (1 for input and 0 for output)
; 1110 0000

      movlw       0xE0
      movwf       TRISD
      banksel     PORTD
      bcf         PORTD,0x00
      bcf         PORTD,0x01

;RD4 - TXPin (out)  RD5 - RXPin (in)

;TEXT MODE is DEFAULT MODE

;Default mode of RC8650 is Text mode
;So keep this
;Change the volume by nV command
;n = [0,9] with 5 as default
;Change to 4
      movlw       CTRLA
      call        TXSW
      movlw       '4'
      call        TXSW
      movlw       'V'
      call        TXSW

      movlw       0x30
      movwf       VOICE       ;starting from 0



BEGIN
      banksel     RCSreg
      clrf        RCSreg
;Change the Voice to nO command
;0 for Perfect Paul (Default)
;1 for Vader
;2 for Big Bob
;3 for Precise Pete
;4 for Ricochet Randy
;5 for Biff
;6 for Skip
;7 for Robo Robert
;
;Apply all 8 voices one at a time
      movlw       CTRLA
      call        TXSW
      movf        VOICE,0           ;write the VOICE
      call        TXSW
      movlw       'O'
```

```
      call        TXSW
;
      call        Dialing
      call        delay1s
      call        delay1s

;Text Message

      call        Message
      call        delay1s
      call        delay1s



;next voice
      incf        VOICE
      btfss       VOICE,0x03  ;third bit =1 means VOICE=8
      goto        BEGIN
      movlw       0x30
      movwf       VOICE       ;again with 0
      call        delay1s
      call        delay1s
      goto        BEGIN

;=== Phone Number Table=
PhoneTable
      movf        PHONEdigit,0
      addwf       PCL
                              ;PC+0
      DT          "12028064821"     ;11 numbers
      retlw 0
;
MessageTable
      movf        MESSAGEdigit,0
      addwf       PCL
      DT          "I LOVE YOU!"     ;11 texts
      retlw       0
;
;Subroutine Dialing

;DTMF Generation (command is n*)
;Call the following Number
;1-202-806-4821
;DTMPF usual (default) burst duration is 100ms
;this could become 500ms by changing the Protocol Options Register
;by nG command
; CTRLA>"1">"6">"0">"G" would change it to 500ms
;Fro details see the RC8650 data sheet
Dialing
      movlw       0x0B        ;11 phone digits
      movwf       Ptemp
      clrf        PHONEdigit
Dagain
      movlw       CTRLA
      call        TXSW
      call        PhoneTable
```

```
      call        TXSW
      movlw       '*'
      call        TXSW
      incf        PHONEdigit
      decfsz      Ptemp
      goto        Dagain
      return
;
;Subroutine Message
Message
      movlw       0x0B        ;11 characters
      movwf       Mtemp
      clrf        MESSAGEdigit
Magain
      call        MessageTable
      call        TXSW
      incf        MESSAGEdigit
      decfsz      Mtemp
      goto        Magain
      movlw       0x0D
      call        TXSW        ;CR key for voicing
      return

;Software TX routine
;The data to be sent is stored in W
TXSW
      banksel     RCSreg
      movwf       RCSreg
      movlw       0x08        ;8 --->W
      movwf       Bitcount    ;8 data bits

;send a START bit
      bcf         PORTD, TXPin
;delay for 1*(BC) cycles
      call  Delay52us   ;Keep this!
TXNEXT
      bcf         STATUS, CARRY
      rrf         RCSreg            ;LSB first mode (normal)
      btfsc       STATUS,CARRY
      bsf         PORTD, TXPin
      btfss       STATUS,CARRY
      bcf         PORTD, TXPin
      call        Delay52us
      decfsz      Bitcount
      goto        TXNEXT
;send STOP bit
      bsf         PORTD, TXPin
      call        Delay52us         ;
;wait until the end of STOP bit
      return


;
;===SUBROUTINES ====
;delay 52us for one baud cycle of 19200 bps
Delay52us
      movlw       0x54
```

```
      movwf       Kount52us
R52us decfsz      Kount52us
      goto        R52us
      return

;=========================================================
;DELAY SUBROUTINES
;
;100us delay needs 500 instruction cycles
;  500 =166*3 +2 ---->Kount=166=0xA6
;  or  =165*3 +5 ---->Kount=165=0xA5
;  or  =164*3 +8 ---->Kount=164=0xA4
Delay100us
      banksel     Kount100us
      movlw       H'A4'
      movwf       Kount100us
R100us
      decfsz      Kount100us
      goto        R100us
      return

;
;10ms delay
; call 100 times of 100 us delay  (with some time discrepancy)
Delay10ms
      banksel     Kount10ms
      movlw       H'64' ;100
      movwf       Kount10ms
R10ms call        delay100us
      decfsz      Kount10ms
      goto        R10ms
      return
;
;1 sec delay
;call 100 times of 10ms delay
Delay1s
      banksel     Kount1s
      movlw       H'64'
      movwf       Kount1s
R1s   call        Delay10ms
      decfsz      Kount1s
      goto        R1s
      return
;
;END OF CODE
        END
```

3. Coding for a Complete System of Voice Synthesizer, LCD, and Keyboard

Now, it's about time to connect the keyboard, the LCD module, and the RC8650 evaluation system for the final version of this application. Our scheme here is that the keyed characters are displayed to the LCD module and that the texts are pronounced as text message when CR key is entered. As we know, the CR key also moves the cursor to the first position of the next line of

the current cursor position. The schematic for the connections and pin assignments are as shown below.

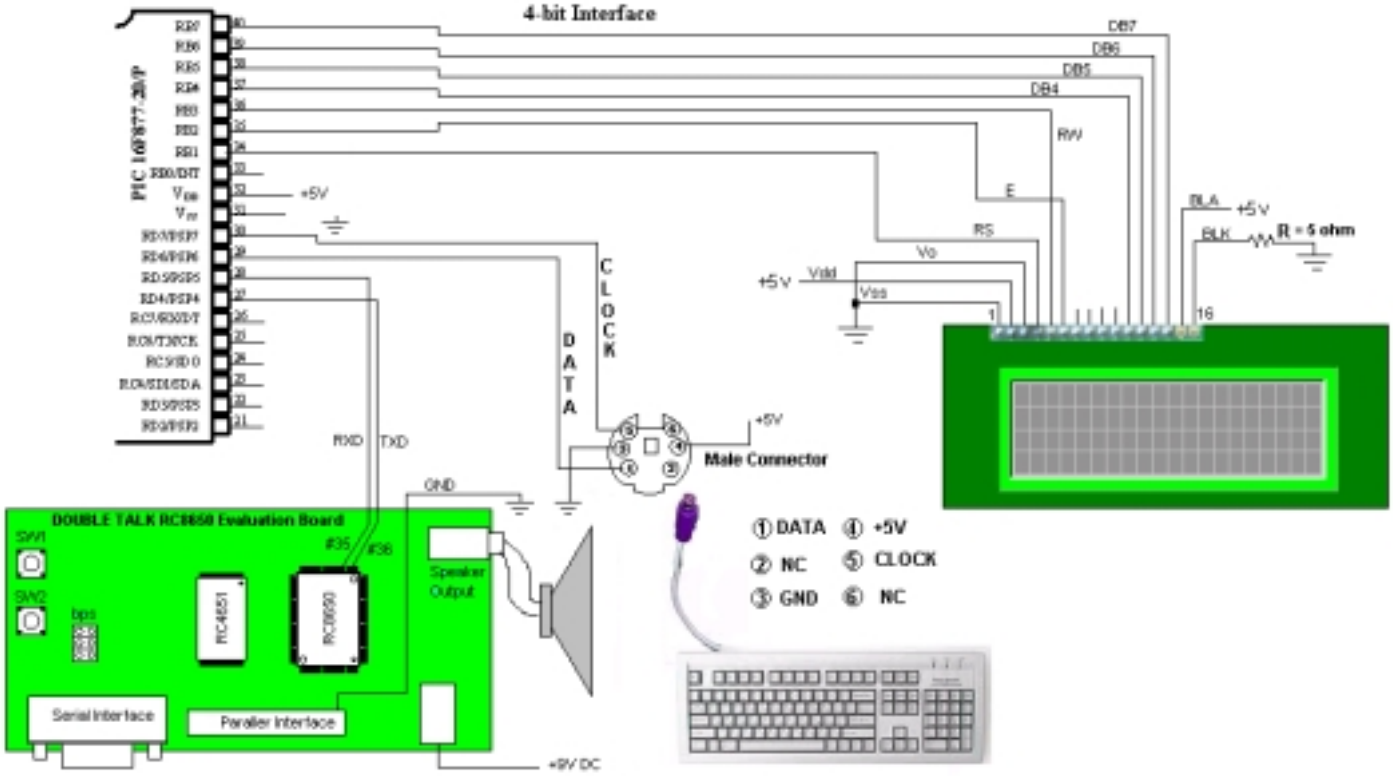

Fig. 83 Keyboard and LCD module connection to RC8650 evaluation system

Before we examine an example code for this final version, let's consider some new stuffs we bring to the project compared with the previous version. First, we have to have a storage space to store texts so that, when CR is pressed, we retrieve them and write to the RC8650 chip set for text-to-voice conversion. Second, we have to accommodate the BS key. When BS key is pressed, we have to not only move the cursor on LCD module to move back by one space but also change the stored text so that the keys after the BS overrides the text previously entered in to a storage location.

Let's discuss about saving the entered characters. Since we have general purpose register spaces in Bank 1 (Bank 0 spaces are usually occupied by the variables defined in the program.) of the RAM, we will going to use the first 80 bytes of the free space in Bank 1. Since we have 4x20 (total of 80 characters), the first 80 spaces, starting from address A0h, completely fit to our purpose. For this allocation, we will use an indirect addressing mode by using INDF and FSR registers. INDF register is the indirect file register to hold a byte data and FSR is the file register selection register. The content of FSR is the address of INDF. INDF is not a physical register and FSR is the address pointer for INDF. In other words, if FSR contains A0h, and if you have the following instruction:

```
movlw       'A'
movwf       INDF
```

Then, the hex number 41h (for 'A') would be written to the address A0h indicated by FSR. If you want to write 'B' at the address A1h, you increase FSR by 1 and write it to INDF:

```
      incf        FSR
      movlw       'B'
      movwf       INDF
```

So, as we wee here, there is no direct contact or control with INDF register, instead, they are performed via FSR.

In example code, we store any character to the storage space staring at A0h until we receive a CR key. When CR is entered, we send the whole text to RC8650 chip set followed by CR (which triggers the text to voice conversion). After the conversion, we move the FSR to the original A0h address so that the next can be overwritten.

So, whenever a text (other than CR or BS) is entered, it is interpreted to ASCII character using the table of `NoSHiftKeyTable, ShiftKeyTable`, or `CAPKeyTable`, depending upon the pressing of Shift or Caps Lock key, or not. Then, it is displayed on the LCD module, and at the same time, the storage address for the text is increased by 1 and the count for number of texts entered is also increased by 1. The variable `Nchar` in the instruction below monitors the number of texts stored for the voice conversion.

An example code when no Shift or Caps Lock key is pressed.

```
      call        NoShiftKeyTable         ;(X) display
      movwf       INDF              ;store the character at INDF
      incf        FSR
      incf        Nchar
      call        LCDisplay
```

An example code when Caps Lock key is pressed.

```
      call        CAPKeyTable
      movwf       INDF              ;store the character at INDF
      incf        FSR
      incf        Nchar
      call        LCDisplay
```

The subroutine for converting the stored texts into voice is shown below. The subroutine `VoiceText` changes the RC8650 mode to text mode (this is not required since the default mode is the text mode), then write the stored texts as numbered by the content of `Nchar` in the software implemented, asynchronous serial communication routine `TXSW`. After writing all the texts stored, it then sends CR to trigger the RC8650 to convert the texts to voice. Once the job is done, it moves the storage starting address to A0h.

```
;subroutine
VoiceText
;if Nchar=0 return (nothing to display)
      movlw       CTRLA
      call        TXSW
      movlw       'T'
      call        TXSW        ;text mode
```

```
      banksel     Nchar
      movf        Nchar,0
      clrf        STATUS
      xorlw       0x00
      btfsc       STATUS,ZERO
      return
      movlw       0xA0
      movwf       FSR
NextChar
      movf        INDF,0
      call        TXSW
      decfsz      Nchar
      goto        domore
      movlw       0x0D        ;voice ON
      call        TXSW
      movlw       0xA0
      movwf       FSR
      return
DOmore
      incf        FSR
      goto        NextChar
```

When BS key in entered, the address pointer for the text must be decreased by 1 and the number of texts must be also decreased by 1:

```
      decf        FSR
      decf        Nchar
```

Even though we discussed in detail about the LCD display of the keyboard keys, since this final version involves very important part of voice conversion, the following code lists the full program for the connection of a AT or PS/2 type keyboard, an 20x4 LCD module with 4-bit interfacing scheme, and a voice synthesize chip set of RC8650. Check for the changes in the PCLATH related instruction at the tables.

```
;kbd6.asm
;
;kbd-LCD-Voice Synthesizer connection
;CR key will start the text-voice conversion
;
;texts tpyed are stored at the RAM space (bank 1) starting @A0
;using FSR pointer and INDF register
;FSR directs the INDF register
;When CR is entered, then the text-voice routine is called
;to make sounds.
;
;   Baud rate for this is set as 19200 for SW enabled Serial Communication
;   to RC8650 Chip Set
;
;This program is asynchronous communication using software method
```

```
;
;F = 20 MHz
;B = Baud Rate
;For B=19200, one Baud cycle (BC) is about 52uS
;
;
;TRANSMIT MODE
;First START bit is sent by setting the TX pin to LOW for (BC) seconds
;And, from then on, the TX Pin is Set/Cleared corresponding to the data bit
;every (BC) seconds.
;8N1 format
;
;TX Pin = RD4
;RX Pin = RD5
;
; LCD is with 4-bit interfacing
;
;CR key would change the line
;
; Pin Connection from LCD to 16F877
; LCD (pin#)      16F877 (pin#)
;DB7 (14) -----RB7(40)
;DB6 (13) ----RB6(39)
;DB5 (12) ----RB5(38)
;DB4 (11) ----RB4(37)
;E (6)  ------RB2(35)
;RW (5)  -----RB3(36)
;RS (4)  -----RB1(24)
;Vo (3)  -----GND
;Vdd (2) ----+5V
;Vss (1) -----GND
;
;KEYBOARD Interfacing
;CLOCK -----RD7 (input)
;DATA ------RD6 (input)
;
;==RC8650 pin connection ==
;RD4 - TXPin (out)
;RD5 - RXPin (in)

      list P = 16F877
INDF        EQU   0x00        ;indirect register
FSR         EQU   0x04        ;Pinter of INDF
STATUS      EQU   0x03
PCL         EQU   0x02        ;For Key Table Calling
PCLATH      EQU   0x0A        ;upper part of PC
CARRY       EQU   0x00
BORROW      EQU   0x00
ZERO        EQU   0x02
PORTB       EQU   0x06
TRISB       EQU   0x86
RS          EQU   0x01        ;RB1
E           EQU   0x02        ;RB2
RW          EQU   0x03        ;RB3
TRISD       EQU   0x88
PORTD       EQU   0x08
CARRY       EQU   0x00
```

```
MSB         EQU   0x07
CLOCK       EQU   0x07        ;from Keyboard (RD7)
KDATA       EQU   0x06        ;from Keyboard (RD6)
TXPIN       EQU   0x04        ;to RC8650
RXPIN       EQU   0x05        ;RD5
MSB         EQU   0x07
CTRLA       EQU   0x01        ;RC8650 Command Character


;RAM Area

      CBLOCK      0x20
            RCSreg                  ;RC8650
            VOICE
;
            CURSOR            ;tracking the current display position
;CURSOR (tracking purpose) (Decimal)
;1  2  3   4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20  line1
;21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40  line 2
;41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60  line 3
;61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80  line 4
;
;DDADDR CONTENT read from LCD Module (HEX)
;00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13  line1
;40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F 50 51 52 53  line 2
;14 15 16 17 18 19 1A 1B 1C 1D 1E 1F 20 21 22 23 24 25 26 27  line 3
;54 55 56 57 58 59 5A 5B 5C 5D 5E 5F 60 61 62 63 64 65 66 67  line 3

            DDaddr            ;Display address (cursor pos)
            DDtemp1
            DDtemp2
            Nchar

            Dkey        ;Key character to be displayed
            DATAreg
            Bitcount
            Kstat
            Kount52us
            Kount120us   ;Delay count (number of instr cycles for delay)
            Kount100us
            Kount1ms
            Kount10ms
            Kount100ms
            Kount1s
            Kount10s
            Kount1m
            Temp        ;temp storage
      ENDC

;program should start from 0005h
;0004h is allocated to interrupt handler

      org         0x0000
      goto        START


      org         0x05
```

```
Start

      banksel     TRISD
; 1100 0000

      movlw       B'11100000' ;Rd7 for CLOCK and Rd6 for DATA as inputs
                              ;rd5 as RX from RC8600
                              ;rd4 as TX to RC8600
      movwf       TRISD

      call        delay1s           ;Give Keyboard to send STATUS to the host

      BANKSEL     TRISB
      movlw       0x00
      movwf       TRISB       ;All output

      banksel     PORTB
      clrf        PORTB       ;RW set LOW here

      clrf        CURSOR            ;Current Display Location
      incf        CURSOR            ;Home cursor position (1, 1)
;LCD routine starts
      call        delay10ms
      call        delay10ms

      banksel     PORTB
      clrf        PORTB       ;RW set LOW here
                              ;give LCD module to reset automatically
      call        LCD4INIT

;END OF LCD INITIALIZATION
;RC8600 setup
;TEXT MODE is DEFAULT MODE

;Default mode of RC8650 is Text mode
;So keep this
;Change the volume by nV command
;n = [0,9] with 5 as default
;Change to 6
      movlw       CTRLA
      call        TXSW
      movlw       '6'
      call        TXSW
      movlw       'V'
      call        TXSW
;VOICE TYpE SELECTION
;Change the Voice to nO command
;0 for Perfect Paul (Default)
;1 for Vader
;2 for Big Bob
;3 for Precise Pete
;4 for Ricochet Randy
;5 for Biff
;6 for Skip
;7 for Robo Robert

      movlw       CTRLA
```

```
      call        TXSW
      movwf       '1'         ;perfect Paul
      call        TXSW
      movlw       'O'
      call        TXSW
;
      movlw       0xA0
      movwf       FSR         ;data pointer @A0 in Bank 1
;receiving data
      banksel     Nchar
      clrf        NCHAR       ;number of data entered
;=============================================================
;KBD Monitoring
BEGIN

      banksel     DATAreg
      clrf        DATAreg
      clrf        DDADDR            ;DD RAM ADDRESS READ from LCD
;
; CHECK IF THE CLOCK is HIGH at least for 10mS

      banksel     PORTD
      btfss       PORTD, CLOCK
      goto        BEGIN       ;if CLOCK is LOW, start again
      call        Delay10ms   ;10mS delays

;check again for CLCOK
      btfss       PORTD, CLOCK
      goto        BEGIN
;READY FOR CLOCK PULSES

      clrf        KSTAT
KEYIN
;X reading
      call        RX11bit                ;
      clrf        STATUS
      movf        DATAreg,0              ;Break Code?
      xorlw       0xF0
      btfss       STATUS,ZERO
      goto        CAT
;BREAK is detected. Abort It. Resume It
      goto        BEGIN
;Category detection
CAT   clrf        STATUS
      movf        DATAreg,0
      xorlw       0xE0
      btfsc       STATUS,ZERO
      goto        Begin             ;E0 keys (CAT2) are ignored
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0x12
      btfsc       STATUS,ZERO
      goto        LRSHIFT
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0x59
      btfsc       STATUS,ZERO
```

```
      goto        LRSHIFT
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0x58        ;CAPS LOCK
      btfsc       STATUS,ZERO
      goto        CAPS
      movf        DATAreg,0
      clrf        STATUS            ;CR check
      xorlw       0x5A
      btfsc       STATUS,ZERO
      goto        CRhandle
      movf        DATAreg,0
      clrf        STATUS
      xorlw       0x66
      btfsc       STATUS,ZERO
      goto        BShandle    ;Back Space Handling
                              ;L Shift ===>12 | F0 12
                              ;R Shift ===>59 | F0 59
;CAT1 has the format of (X)|(F0)(X)
CAT1  movf  DATAreg,0
;check if the key in is CR
;Then we have to move the next line

      call        NoShiftKeyTable         ;(X) display
      movwf       INDF                    ;store the character at INDF
      incf        FSR
      incf        Nchar
      call        LCDisplay
;(F0) detection
      call        RX11bit
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0xF0
      btfss       STATUS,ZERO
;Key is not broken. Still pressed,
      goto        CAT1
;Key is broken
;Last (X) reading
      call        RX11bit                 ;(X) after F0

      goto        BEGIN

;L-SHIT and R-SHIFT has the form
;L-SHIFT and a character 12 X | F0 X |F0 12
;R-SHIFT and a character 59 X | F0 X |F0 59

LRSHIFT                           ;12 or 59 entered

;(F0) detection
      call        RX11bit
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0xF0
      btfsc       STATUS,ZERO
      goto        BEGIN

                              ;X
```

```
      clrf        STATUS            ;if (12) do not display
      movf        DATAreg,0
      xorlw       0x12
      btfsc       STATUS, ZERO
      goto        LRSHIFT

      clrf        STATUS            ;if (59) do not display
      movf        DATAreg,0
      xorlw       0x59
      btfsc       STATUS, ZERO
      goto        LRSHIFT

                        ;a Key (X) is entered
      movf        DATAreg,0
      call        ShiftKeyTable
      movwf       INDF              ;store the character at INDF
      incf        FSR
      incf        Nchar
      call        LCDisplay


;(F0) detection
      call        RX11bit
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0xF0
      btfss       STATUS,ZERO
      goto        LRSHIFT
;Last (X) reading
      call        RX11bit
      movf        DATAreg,0
      clrf        STATUS            ;check if (X) or (12) entered after F0
      xorlw       0x12
      btfsc       STATUS,ZERO
      goto        BEGIN
      goto        LRSHIFT
;
CAPS                          ;caps lock (58) entered

;(F0) detection
      call        RX11bit                ;this must be F0
      call        RX11bit                ;this must be (58) again

CAPnext
      call        RX11bit                ;Check if (58) or other
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0x58
      btfss       STATUS,ZERO
      goto        CAPtwo                 ;End of CAP session
      call        RX11bit                ;F0
      call        RX11bit                ;(58)
      goto        BEGIN
                              ;a Key (X) is entered
CAPtwo
      movf        DATAreg,0
```

```
      call        CAPKeyTable
      movwf       INDF              ;store the character at INDF
      incf        FSR
      incf        Nchar
      call        LCDisplay

;(F0) detection
      call        RX11bit                 ;this
      clrf        STATUS
      movf        DATAreg,0
      xorlw       0xF0
      btfss       STATUS,ZERO
      goto        CAPtwo


;Last (X) reading             ;F0 is read
      call        RX11bit                 ;(X) again and ignore
      goto        CAPnext


;CR handling
CRhandle
      call        RX11bit           ;F0 read
      call        RX11bit           ;CR reading again
; text-voice conversion
      call        VoiceText
;
;read the current cursor position
      call        readad4
;DDADDR has the content
;NOTE: MSB must be 1 in the cursor command
      bsf         DDADDR, MSB
;if DDADDR<94, then new cursor position is C0
;if DDADDR<E8, then 80
;if DDADDR<C0, then D4
;if DDADDR<D4, then 94
      clrf        STATUS
      movf        DDADDR,0
      sublw       0x94        ;k-W -->W
      btfsc       STATUS,Borrow     ;No borrow means that k>W
      goto        CR94        ; is less than 94
      clrf        STATUS
      movf        DDADDR,0
      sublw       0xC0
      btfsc       STATUS,Borrow
      goto        CRC0

      clrf        STATUS
      movf        DDADDR,0
      sublw       0xD4
      btfsc       STATUS,Borrow
      goto        CRD4

      clrf        STATUS
      movf        DDADDR,0
      sublw       0xE8
      btfsc       STATUS,Borrow
```

```
      goto        CRE8
      goto        BEGIN

CR94  call        posline12
      goto        begin

CRC0  call        posline14
      goto        BEGIN

CRD4  call        posline13
      goto        BEGIN


CRE8  call        LCDclearhome                  ;clear screen first
      call        posline11
      goto        BEGIN

;CRE8 call        posline11
;     goto        BEGIN

;BS Handling
BShandle
      movf        DATAreg,0   ;W holds $66
      call        RX11bit           ;F0 read
      call        RX11bit           ;BS break code
;read the current cursor position
      call        readad4
;DDADDR has the content
; SO move the current to the left
;NOTE: MSB must be 1 for commanding of the cursor position
      bsf         DDADDR, MSB
;if DDADDR = 94, then new cursor position is D3
;if DDADDR = C0, then new position is 93
;if DDADDR = D4, then new position is A7
;if DDADDR = 80, then new position is 80 (NO CHANGE)
; all other cases, new position is (DDADDR - 1)
      clrf        STATUS
      movf        DDADDR,0
      xorlw       0x94
      btfsc       STATUS, ZERO
      goto        DD94
      clrf        STATUS
      movf        DDADDR,0
      xorlw       0xC0
      btfsc       STATUS,ZERO
      goto        DDC0
      clrf        STATUS
      movf        DDADDR,0
      xorlw       0xD4
      btfsc       STATUS,ZERO
      goto        DDD4
      clrf        STATUS
      movf        DDADDR,0
      xorlw       0x80
      btfsc       STATUS,ZERO
      goto        DD80
;all others
```

```
      decf        DDADDR
      decf        CURSOR
      movf        DDADDR,0
      call        instw4
      decf        FSR
      decf        Nchar
      goto        BEGIN

DD94  movlw       0xD3
      decf        CURSOR
      call        instw4
      decf        FSR
      decf        Nchar
      goto        BEGIN

DDC0  movlw       0x93
      decf        CURSOR
      call        instw4
      decf        FSR
      decf        Nchar
      goto        BEGIN

DDD4  movlw       0xA7
      decf        CURSOR
      call        instw4
      decf        FSR
      decf        Nchar
      goto        BEGIN

DD80  movlw       0x80
      call        instw4
      decf        FSR
      decf        Nchar
      goto        BEGIN


;

;=====================================================
;SUBROUTINE LCD4INIT
;Function for 4-bit (only one write must be done)
;In other words, send only the high nibble
LCD4INIT
;IMPORTANT PART
      movlw       0x28
      call        hnibble4
;Function for 4-bit, 2-line display, and 5x8 dot matrix
      movlw       0x28
      call        instw4
;Display On, CUrsor On, No blinking
      movlw       0x0E        ;0F would blink
      call        instw4
;DDRAM address increment by one & cursor shift to right
      movlw       0x06
      call        instw4
;DISPLAY CLEAR
CLEAR
```

```
      movlw       0x01
      call        instw4
;
      call        posline11         ;pos1 and line 1
;now CURSOR=1
      return
;================================================

;LCD DISPLAYING SUBROUTINE
LCDisplay

      call        dataw4
      incf        CURSOR            ;every time of display, increase cursor
;CURSOR is automatically incremented by 1 from LCDisplay
;if CURSOR is 20 (0x14), change to posline12
;if CURSOR is 40 (0x28), change to posline13
;if CURSOR is 60 (0x3C), change to posline14
;if CURSOR is 80 (0x50), change to posline11
      clrf        STATUS
      movf        CURSOR,0
      xorlw       0x15
      btfsc       STATUS, ZERO
      goto        Toline2

      clrf        STATUS
      movf        CURSOR,0
      xorlw       0x29
      btfsc       STATUS,ZERO
      goto        Toline3

      clrf        STATUS
      movf        CURSOR,0
      xorlw       0x3D
      btfsc       STATUS,ZERO
      goto        Toline4

      clrf        STATUS
      movf        CURSOR,0
      xorlw       0x51
      btfsc       STATUS,ZERO
      call        LCDClearhome            ;delete all and move to (1,1)
      return

Toline2
      call        posline12
      return
Toline3
      call        posline13
      return
Toline4
      call        posline14
      return

;SUBROUTINE
;DISPLAY CLEAR and Cursor to Home position (line 1, position 1)
LCDclearhome
      movlw       0x01
```

```
      call        instw4
;Now let's move the cursor to the home position (position 1 of line #1)
;and set the DDRAM address to 0.  This is done by the "return home"
instruction.

      movlw       0x02
      call        instw4
;home position
      movlw       0x80
      call        instw4
      movlw       0x01
      movwf       CURSOR
      return



;====SUBROUTINES =====
posline11
;Position to pos 1 and line 1
      movlw       0x80
      call        instw4
      movlw       0x01
      movwf       CURSOR
      return

posline12         ;pos 1 and line 2
      movlw       0xC0
      call        instw4
      movlw       0x15  ;21
      movwf       CURSOR
      return

posline13         ;pos1 and line3
      movlw       0x94
      call        instw4
      movlw       0x29  ;41
      movwf       CURSOR
      return
posline14         ;pos 1 and line 4
      movlw       0xD4
      call        instw4
      movlw       0x3D  ;61
      movwf       CURSOR
      return
;
;high nibble only write for the first step of 4-bit set up
hnibble4
      movwf       Temp        ;Temp storage
      movf        Temp,0            ;Now W also holds the data
      andlw       0xF0           ; get upper nibble
      movwf       PORTB         ; send data to lcd
      call        delay1ms
      bcf         PORTB, RS
      call        delay1ms
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E
```

```
      call        delay10ms         ;end of high nibble for 4-bit setup
      return
;
;subroutine instw (4-bit interface instruction write)
;instruction to be written is stored in W before the call
instw4
      movwf       Temp        ;Temp storage
      movf        Temp,0      ;Now W also holds the data
      andlw       0xF0            ; get upper nibble
      movwf       PORTB          ; send data to lcd
      call        delay1ms
      bcf         PORTB, RS
      call        delay1ms
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E
      call        delay10ms         ;end of higher nibble
      swapf       Temp,0            ;get lower nibble to W
      andlw       0xf0
      movwf       PORTB             ;Write to LCD
      call        delay1ms
      bcf         PORTB, RS
      call        delay1ms
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E          ;end of lower nibble
      call        delay10ms
      return

;subroutine dataw (4-bit interface data write)

dataw4
      movwf       Temp        ;Temp storage
      movf        Temp,0      ;Now W also holds the data
      andlw       0xF0            ; get upper nibble
      movwf       PORTB          ; send data to lcd
      call        delay1ms
      bsf         PORTB, RS
      call        delay1ms
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E
      call        delay10ms         ;end of higher nibble
      swapf       Temp,0            ;get lower nibble to W
      andlw       0xF0
      movwf       PORTB             ;Write to LCD
      call        delay1ms
      bsf         PORTB, RS
      call        delay1ms
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E          ;end of lower nibble
      call        delay10ms
      return
;
;subroutine reading the cursor position
;RW Must be High
```

```
;RS Must be Low
;the 7th bit is BF flag (so ignire this one, or make MSB 0)
;PORTB <7:4> as inputs
;High then Low nibbles of ADDRESS
;The content of DDADDR read from LCD module (HEX Numbers)
;Line 1: 00 01 02 ..............  13
;Line 2: 40 41 42 ..............  53
;Line 3: 14 15 16 ..............  27
;Line 4: 54 55 56 ..............  67


readad4
      banksel     TRISB       ;set Rb7 - DR4 as inputs
      movlw       0xF0        ;upper 4 bits as inputs
      movwf       TRISB
      banksel     PORTB
      bsf         PORTB, RW   ;READING MODE
      call        delay1ms
      bcf         PORTB,RS
      call        delay1ms
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E    ;Reading starts here now
                              ;upper byte first

      movlw       0xF0
      andwf       PORTB,0
      movwf       DDtemp1

      bcf         PORTB,RS
      call        delay1ms
      bsf         PORTB, E
      call        delay1ms
      bcf         PORTB, E
                        ;reading starts now
                        ;for lower byte
      movlw       0xF0
      andwf       PORTB,0
      movwf       DDtemp2
      swapf       DDtemp2

;add DDtemp1 and DDtemp2 for DDADDR
;
      movf        DDtemp1,0
      addwf       DDtemp2,0
      movwf       DDADDR            ;The DD Ram ADDRESS

      banksel     TRISB
      movlw       0x00
      movwf       TRISB       ;all outputs again
      banksel     PORTB
      bcf         PORTB,RW    ;back to writing mode
      return
;===============================================
;subroutine
VoiceText
;if Nchar=0 return (nothing to display)
```

```
      movlw       CTRLA
      call        TXSW
      movlw       'T'
      call        TXSW        ;text mode

      banksel     Nchar
      movf        Nchar,0
      clrf        STATUS
      xorlw       0x00
      btfsc       STATUS,ZERO
      return
      movlw       0xA0
      movwf       FSR
NextChar
      movf        INDF,0
      call        TXSW
      decfsz      Nchar
      goto        domore
      movlw       0x0D        ;voice triggered
      call        TXSW
      movlw       0xA0
      movwf       FSR
      return
DOmore
      incf        FSR
      goto        NextChar

;=========================================================
;DELAY SUBROUTINES

Delay120us
      banksel     Kount120us
      movlw       H'C5'       ;D'197'
      movwf       Kount120us
R120us
      decfsz      Kount120us
      goto        R120us
      return
;
Delay100us
      banksel     Kount100us
      movlw       H'A4'
      movwf       Kount100us
R100us
      decfsz      Kount100us
      goto        R100us
      return
;
;1ms delay
Delay1ms
      banksel     Kount1ms
      movlw       0x0A  ;10
      movwf       Kount1ms
R1ms  call        delay100us
      decfsz      Kount1ms
      goto        R1ms
      return
```

```
;
;10ms delay
; call 100 times of 100 us delay  (with some time discrepancy)
Delay10ms
      banksel     Kount10ms
      movlw       H'64' ;100
      movwf       Kount10ms
R10ms call        delay100us
      decfsz      Kount10ms
      goto        R10ms
      return
;
;

;1 sec delay
;call 100 times of 10ms delay
Delay1s
      banksel     Kount1s
      movlw       H'64'
      movwf       Kount1s
R1s   call        Delay10ms
      decfsz      Kount1s
      goto        R1s
      return
;
;
;
;SUBROUTINE RX11bit
;RX Routine for 11-bit read
;1 Start
;8 Data (LSB first)
;1 Parity (Odd)
;1 Stop (HIGH)
;KSTAT Bit Info
; KSTAT<0> : parity
; KSTAT<2>:kBD Error
RX11bit
      clrf        DATAreg
      banksel     PORTD
;Let it have at least 500us CLOCK high period
      btfss       PORTD, CLOCK
      goto        RX11bit           ;if CLOCK is LOW, start again
      call        Delay100us  ;200uS delays
      call        DElay100us
;check again for CLCOK
      btfss       PORTD, CLOCK
      goto        RX11bit
;Clock Check
Scheck
      btfsc       PORTD,CLOCK
      goto        Scheck
      call        delay5us    ;wait for 5us     for data stabilization
      btfsc       PORTD, KDATA
      goto        KERROR                  ;if START BIT is not Zero ERROR
;START Detected
;8-bit Data Check
      movlw       0x08
```

```
      movwf       Bitcount    ;8 data bits
RXNEXT
      bcf         STATUS, CARRY     ;Clear the Carry Bit
      rrf         DATAreg           ;rotate to the right
CKHIGH
      btfss       PORTD, CLOCK      ;Wait for CLOCK to back to High
      goto        CKHIGH
CKLOW btfsc       PORTD, CLOCK      ;wait for CLOCK now to LOW
      goto        CKLOW
      call        delay5us    ;5us delay
      btfsc       PORTD, KDATA      ;0 or 1
      bsf         DATAreg, MSB      ;1? Then set the MSB
      decfsz      Bitcount
      goto        RXNEXT
;Check for Parity Bit
;Wait for CLOCK bacj to High
CKHIGH2
      btfss       PORTD, CLOCK      ;Wait for CLOCK to back to High
      goto        CKHIGH2
CKLOW2
      btfsc       PORTD, CLOCK      ;wait for CLOCK now to LOW
      goto        CKLOW2
      call        delay5us    ;5us delay
      btfsc       PORTD, KDATA      ;Parity Bit
      goto        OneP        ;Pbit=1
      bcf         Kstat,0x00  ;Pbit=0
      goto        Stopcheck
Onep  bsf         Kstat, 0x00 ;Pbit=1
Stopcheck
;wait for CLOCK back to High
CKHIGH3
      btfss       PORTD, CLOCK      ;Wait for CLOCK to back to High
      goto        CKHIGH3
CKLOW3
      btfsc       PORTD, CLOCK      ;wait for CLOCK now to LOW
      goto        CKLOW3
      call        delay5us    ;5us delay
      btfss       PORTD, KDATA      ;STOP bit
      goto        KERROR            ;if STOP=0 , ERROR
      return
KERROR
      bsf         KSTAT, 0x02
      return
;=======================================================
;Software TX routine for RC8650
;The data to be sent is stored in W
TXSW
      banksel     RCSreg
      movwf       RCSreg
      movlw       0x08        ;8 --->W
      movwf       Bitcount    ;8 data bits

;send a START bit
      bcf         PORTD, TXPin
;delay for 1*(BC) cycles
      call        Delay52us   ;Keep this!
TXNEXT
```

```
      bcf         STATUS, CARRY
      rrf         RCSreg            ;LSB first mode (normal)
      btfsc       STATUS,CARRY
      bsf         PORTD, TXPin
      btfss       STATUS,CARRY
      bcf         PORTD, TXPin
      call        Delay52us         ;KEEP THIS!
      decfsz      Bitcount
      goto        TXNEXT
;send STOP bit
      bsf         PORTD, TXPin
      call        Delay52us         ;keep tHIS!
;wait until the end of STOP bit
      return


delay5us
;need total 10 instructions
      nop
      nop
      nop
      nop
      nop
      nop
      nop
      nop
      nop
      return

;delay 52us for one baud cycle of 19200 bps
Delay52us
      movlw       0x54
      movwf       Kount52us
R52us decfsz      Kount52us
      goto        R52us
      return


;100ms delay

Delay100ms
      banksel     Kount100ms
      movlw       0x0A  ;10
      movwf       Kount100ms
R100ms
      call        delay10ms
      decfsz      Kount100ms
      goto        R100ms
      return
;
Delay500ms
      call        delay100ms
      call        delay100ms
      call        delay100ms
      call        delay100ms
      call        delay100ms
```

```
      return

;TABLES
;=====================================================
      org         0x0300                        ;So that all the table

;Without Shift (or CAPs Lock) Key Table
NoshiftKeyTable

      bsf         PCLATH, 0x00
      bsf         PCLATH, 0x01
      addwf       PCL
;
      retlw 0     ;PC+0
      retlw 0     ;PC+1
      retlw 0     ;+2
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0     ;+0D
      retlw 0x60  ;+0E MAKE/BREAK= 0E ---->ASCII = 0x60 Apostrophe
      retlw 0     ;+0F
      retlw 0
      retlw 0
      retlw 0
      retlw 0     ;+13
      retlw 0     ;+14
      DT    "q1"  ;+15, 16
      retlw 0     ;+17
      retlw 0
      retlw 0
      DT    "zsaw2"     ;+1A, 1B, 1C, 1D, 1E
      retlw 0     ;+1F
      retlw 0     ;+20
      DT    "cxde43"    ;+21, 22, 23, 24, 25, 26
      retlw 0     ;+27
      retlw 0     ;+28
      retlw ' '   ;+29 Space
      DT    "vftr5"     ;+2A, 2B, 2C, 2D, 2E
      retlw 0     ;+2F
      retlw 0     ;+30
      DT    "nbhgy6"    ;+31, 32, 33, 34,35,36
      retlw 0     ;+37
      retlw 0     ;+38
      retlw 0     ;+39
      DT    "mju78"     ;+3A, 3B, 3C, 3D, 3E
      retlw 0     ;+3F
      retlw 0     ;+40
      DT    ",kio09"    ;+41, 42,43,44,45,46
      retlw 0     ;+47
```

```
      retlw 0     ;+48
      DT    "./l;p-"    ;+49, 4A, 4B, 4C, 4D, 4E
      retlw 0     ;+4F
      retlw 0     ;+50
      retlw 0     ;+51
      retlw 0x27  ;+52  single quote
      retlw 0     ;+53
      DT    "[="  ;+54, 55
      retlw 0     ;+56
      retlw 0     ;+57
      retlw 0     ;+58
      retlw 0     ;+59
      retlw 0x0D  ;+5A  Return
      retlw ']'   ;+5B
      retlw 0     ;+5C
      retlw 0x5C  ;+5D  \
      retlw 0     ;+5E
      retlw 0     ;+5F
      retlw 0     ;+60
      retlw 0     ;+61
      retlw 0     ;+62
      retlw 0     ;+63
      retlw 0     ;+64
      retlw 0     ;+65
      retlw 0x08  ;+66  Backspace


;With Shift  Key Table
shiftKeyTable
      bcf         PCLATH, 0x00
      bsf         PCLATH, 0x01
      addwf       PCL

      retlw 0     ;PC+0
      retlw 0     ;PC+1
      retlw 0     ;+2
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0
      retlw 0     ;+0D
      retlw 0x7E  ;+0E MAKE/BREAK= 0E ---->ASCII 7E (~)
      retlw 0     ;+0F
      retlw 0
      retlw 0
      retlw 0
      retlw 0     ;+13
      retlw 0     ;+14
      DT    "Q!"  ;+15, 16
      retlw 0     ;+17
      retlw 0
```

```
      retlw 0
      DT    "ZSAW@"     ;+1A, 1B, 1C, 1D, 1E
      retlw 0     ;+1F
      retlw 0     ;+20
      DT    "CXDE$#"    ;+21, 22, 23, 24, 25, 26
      retlw 0     ;+27
      retlw 0     ;+28
      retlw ' '   ;+29 Space
      DT    "VFTR%"     ;+2A, 2B, 2C, 2D, 2E
      retlw 0     ;+2F
      retlw 0     ;+30
      DT    "NBHGY^"    ;+31, 32, 33, 34,35,36
      retlw 0     ;+37
      retlw 0     ;+38
      retlw 0     ;+39
      DT    "MJU&*"     ;+3A, 3B, 3C, 3D, 3E
      retlw 0     ;+3F
      retlw 0     ;+40
      DT    "<KIO)("    ;+41, 42,43,44,45,46
      retlw 0     ;+47
      retlw 0     ;+48
      DT    ">?L:P_"    ;+49, 4A, 4B, 4C, 4D, 4E
      retlw 0     ;+4F
      retlw 0     ;+50
      retlw 0     ;+51
      retlw 0x22  ;+52 double quote
      retlw 0     ;+53
      DT    "{+"  ;+54, 55
      retlw 0     ;+56
      retlw 0     ;+57
      retlw 0     ;+58
      retlw 0     ;+59
      retlw 0x0D  ;+5A  Return
      retlw '}'   ;+5B
      retlw 0     ;+5C
      retlw 0x7C  ;+5D  |
      retlw 0     ;+5E
      retlw 0     ;+5F
      retlw 0     ;+60
      retlw 0     ;+61
      retlw 0     ;+62
      retlw 0     ;+63
      retlw 0     ;+64
      retlw 0     ;+65
      retlw 0x08  ;+66  Backspace

;CAPs Lock Key Table
      org         0x0400
CAPKeyTable
      bsf         PCLATH, 0x02
      bcf         PCLATH, 0x01
      bcf         PCLATH, 0x00
      addwf       PCL
      retlw 0     ;PC+0
      retlw 0     ;PC+1
      retlw 0     ;+2
      retlw 0
```

```
retlw 0
retlw 0
retlw 0
retlw 0
retlw 0
retlw 0
retlw 0
retlw 0
retlw 0
retlw 0     ;+0D
retlw 0x60  ;+0E MAKE/BREAK= 0E ---->ASCII = 0x60 Apostrophe
retlw 0     ;+0F
retlw 0
retlw 0
retlw 0
retlw 0     ;+13
retlw 0     ;+14
DT    "Q1"  ;+15, 16
retlw 0     ;+17
retlw 0
retlw 0
DT    "ZSAW2"     ;+1A, 1B, 1C, 1D, 1E
retlw 0     ;+1F
retlw 0     ;+20
DT    "CXDE43"    ;+21, 22, 23, 24, 25, 26
retlw 0     ;+27
retlw 0     ;+28
retlw ' '   ;+29 Space
DT    "VFTR5"     ;+2A, 2B, 2C, 2D, 2E
retlw 0     ;+2F
retlw 0     ;+30
DT    "NBHGY6"    ;+31, 32, 33, 34,35,36
retlw 0     ;+37
retlw 0     ;+38
retlw 0     ;+39
DT    "MJU78"     ;+3A, 3B, 3C, 3D, 3E
retlw 0     ;+3F
retlw 0     ;+40
DT    ",KIO09"    ;+41, 42,43,44,45,46
retlw 0     ;+47
retlw 0     ;+48
DT    "./L;P-"    ;+49, 4A, 4B, 4C, 4D, 4E
retlw 0     ;+4F
retlw 0     ;+50
retlw 0     ;+51
retlw 0x27  ;+52  single quote
retlw 0     ;+53
DT    "[="  ;+54, 55
retlw 0     ;+56
retlw 0     ;+57
retlw 0     ;+58
retlw 0     ;+59
retlw 0x0D  ;+5A  Return
retlw ']'   ;+5B
retlw 0     ;+5C
retlw 0x5C  ;+5D  \
retlw 0     ;+5E
```

```
      retlw 0     ;+5F
      retlw 0     ;+60
      retlw 0     ;+61
      retlw 0     ;+62
      retlw 0     ;+63
      retlw 0     ;+64
      retlw 0     ;+65
      retlw 0x08  ;+66  Backspace

;END OF CODE
        END
```

I hope you have enough patience to learn about this new board and coding for 16F877 chip.