MicroToys Guide: **SpeakJet** K. Lloyd & R. Roy February 2005

## Introduction

This guide shows how to use the SpeakJet voice synthesizer in conjunction with a low power audio amplifier, the Motorola MC34119P. The SpeakJet generates specific parts of words known as phonemes which can be combined to create any word in the English language. The SpeakJet is controlled through a serial connection, using the UART onboard the PIC**.** The output of the SpeakJet is a mere 25mA, which when powering a typical 8 Ohm speaker will be barely audible. To fix this a low powered audio amplifier is used to amplify the output.

**Note:**

- The audio amplifier did work with the particular setup in this documentation though it may not work ideally in other applications
- The SpeakJet was operable when the PIC operated at 1 MHz, otherwise the serial communication did not work properly.

## Connecting the Components

Note: The described connections are all illustrated in Figure 1.

The SpeakJet has two methods of control; 1) real-time serial control and 2) event input line control. In the real-time method all speaking commands are processed by the SpeakJet immediately. This requires that anytime that the SpeakJet talks, the PIC will need to be sending phrases to it. In the event input method, stored phrases are triggered by an event pin. This requires that phrases first be programmed into the SpeakJet's memory. When the phrase needs to be spoken, the corresponding event line simply needs to be set high.

This guide uses the first method, the real-time method, to control the SpeakJet. The only connection between the SpeakJet and the PIC is a single serial connection from the transceiver on the PIC (port RC6/TX) to the receiver on the SpeakJet (RCX). In order to use the SpeakJet to speak it must be put in "normal operation" mode which requires tying M0 low and M1 high. Since event control is not being implemented, pins E0-E7 are tied low.

The audio signal comes out the $V_{out}$ pin on the SpeakJet. This is connected to the audio amplifier through a 0.1 μF ceramic capacitor in series with a resistor (see amplifier section for details on resistor). The capacitor acts as a high pass filter, filtering out low inaudible frequencies.

## Configuring the Serial Interface

The SpeakJet serial interface has a default configuration of 9600 baud, no parity, and 1 stop bit (8, N, 1), so the PIC is configured to match these settings. First set the Transmit Status and Control Register (TXSTA), and enable the Serial Port by setting bit-7 of RCSTA high.

```
      movlw b'00100110'       ; See PIC datasheet p 168
      movwf TXSTA
      movlw b'10000000'       ; Turn serial port on, disable receive.
      movwf RCSTA
```

Then set the Baud rate by setting the SPBRG register equal to the appropriate value (see pg 168-171 of PIC manual for values). For example (assuming High Baud Rate was set in TXSTA and running at 1 MHz):

```
      movlw d'6'        ;129 for a 20 MHz clock yields Baud of 9600
      movwf SPBRG
```

## SpeakJet Phonemes

It is simple to send information from the PIC to the UART, simply move the information that needs to be sent to the TXREG registry.

To get the SpeakJet to speak the following steps must be executed:

1) Issue the command to stop any enunciation
2) Issue the command to clear the buffer
3) Issue the command to start enunciation
4) Send the codes for the phonemes to be spoken

Enunciation is an SCP command (explained bellow) instructing the SpeakJet to start enunciating any phonemes that are in the SpeakJet's 64 byte input buffer.

Sending phonemes to the SpeakJet to be spoken is a simple task; simply send the code for that particular phoneme over the serial line. For example to send the command to speak the "LO" phoneme the following code would work:

```
      movlw       0x92       ; Move the code for LO into the W register
      movwf       TXREG      ; Transmit byte in W
```

In order to access the functions that control other options of the SpeakJet, such as telling it to start enunciating, clear the buffer, access memory, etc, Serial Control Protocol (SCP) must be used. To call a SCP function just send '\X' to the SpeakJet where X is the command (see pg 8 of manual). The following is example code demonstrating how to command the SpeakJet to start enunciating.

```
      movlw '\\'             ;Following 4 lines put the SpeakJet in SCP mode
      call  uart_put         ;
      movlw '0'              ;
      call  uart_put         ;
      movlw 'S'              ;Tell the SpeakJet to start enunciating
      call  uart_put         ;
      movlw 'X'              ;Exit SCP mode
      call  uart_put         ;

uart_put
      btfss       PIR1, TXIF        ; Wait for TXREG to be empty
      goto  uart_put
      movwf       TXREG             ; Transmit byte in W
      return
```

If you notice that in the midst of a long phrase the SpeakJet stops speaking, this is because the buffer is full. This can be fixed by sending part of the phrase, clearing the buffer, and then sending the next part of the phrase. The buffer is only 64 bytes, and the D2 pin on the SpeakJet goes high when the buffer is half full. It is possible to program an interrupt that will stop sending messages and clear the buffer when the D2 pin goes high, however it is not covered in this guide.

## Low Power Audio Amplifier

The amplifier used in this guide has its differential gain defined by Gain=2 x $R_f/R_i$, where $R_i$ is the resistor in series with the audio input and $R_f$ is the resistor in parallel with the amplifier. The recommend values are $R_i = 3.0$ kΩ and $R_f > 30$ kΩ which yield a gain of 10 (the data sheet recommends that the gain be kept below 50). Gains much small than this cause a positive feedback loop from $V_{01}$ to $V_{in}$ completely distorting the sound. The speaker connected to the amplifier will have one terminal connected to $V_{O1}$ and the other to $V_{O2}$.

The FC1 and FC2 pins on the amplifier should be connected to ground through capacitors. Typical values for the capacitors are 1.0 µF and 5.0 µF respectively and their purpose is to filter out any high frequency noise in the power supply.

The CD pin, the Chip Disable pin, simply turns off the amplifier whenever it is set high, and is useful to limit power consumption or as a mute feature.

The power supply for the amplifier, $V_{cc}$, can be the same as the power supply of the SpeakJet and PIC. Manufacturer specifications require operating voltage to be between 2 and 16 Volts DC.

The data sheet for the audio amplifier provides distortion and frequency response graphs for a range of different resistor and voltage parameters.

## Sample Code

The attached sample code goes through an example of how to have the SpeakJet say “Kevin and Raj.” The constants established at the begging of the code are documented in the SpeakJet user manual.

The code essentially has two parts; first initialize the needed registries (serial registries and the timer0 registry) and second send the phonemes to the SpeakJet.

The following Appendix contains the registers that are initialized and what each bit pertains to.

## Appendix A

**T0CON: (Timer0 configuration) Register bit descriptions**

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

**Note:** A prescale value scales the timer0 to count in different amount of delays. For example in our case we chose bit2-0 to be 111, setting the prescale value to be 1:256. This means after 256 clock cycles our timer0 increments by 1. Thus this is the largest delay possible by timer0 in between increments.

**RCSTA: Receive Status and Control Register**
Note: The only reason this register is needed is because it has the Serial Port Enable Bit, all other bits are useless since we do receive any serial communication in this guide.

bit 7: **SPEN**: Serial Port Enable Bit
1 = Serial port enabled
0 = Serial port disabled

bit 6-0: do not matter since not used.

**TXSTA: Tansmit Status and Control Register**
Note: only Asynchronous settings listed, for Synchronous mode see pg 166 of PIC datasheet.

bit 7 **CSRC**: Clock Source Select bit (doesn't matter)
bit 6 **TX9**: 9-Bit Transmit Enable bit
1 = 9-bit transmission
0 = 8-bit transmission
bit 5: **TXEN**: Transmission Enable bit
1 = Enable
0 = Disable
bit 4: **SYNC**: USART Mode Select Bit
0 = Asynchronous mode
bit 3: **UnImplementeed**: Read as '0'
bit 2: **BRGH**: High Baud Rate Select Bit
1 = High Speed
0 = Low Speed
bit 1: **TRMT**: Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full
bit 0: **TX9D**: 9th bit of Transmit Data
Can be Address/Data bit or a parity bit

## Specifications

PIC18CXX2 Data Sheet

http://ww1.microchip.com/downloads/en/DeviceDoc/39026c.pdf

SpeakJet

Operating Voltage: 2-5.5 $V_{dc}$
Output: 25 mA
http://www.magnevation.com/pdfs/speakjetusermanual.pdf

Motorola/Freescale MC34119 Low Power Audio Amp.

Operating Voltage: 2-12 $V_{dc}$
Drive Speaker Impedance: 8-32 Ohms
Output Current: 250 mA
http://www.freescale.com/files/timing_interconnect_access/doc/data_sheet/MC34119.pdf

## Supplier

| Part | Vendor | Part # | Price |
|---|---|---|---|
| Speakjet | SpeechChips.com | SpeakJet | $21.99 |
| Audio Amplifier | DigiKey | MC34119P | $1.53 |

## Additional Resources

Automated Wakeup Call Generator: E155 Final Project

By: Esteban Molina-Estolano and Matt Reynolds
http://odin.ac.hmc.edu/~harris/class/e155/projects04/wakeup.pdf

## Schematics

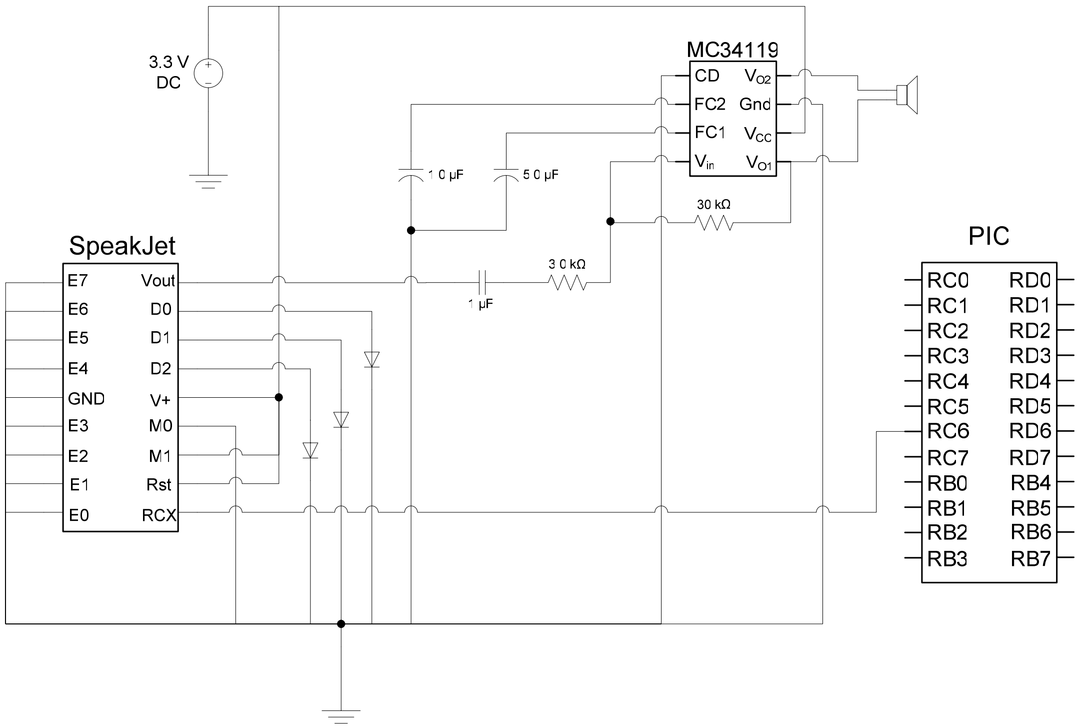

Figure 1. Schematic of the PIC, SpeakJet and amplifier connected.

```
; SpeakJet.asm
; SpeakJet MicroToys Sample Code
; By: Kevin and Raj


; Use the 18F452 PIC microprocessor List p=18f452,f=inhx32
    #include <p18f452.inc>
;
; Overview of Code:
 ; - Declare phoneme constants
;  - Store sentence into databank
;  - Initialize timer and serial registers
;  - Read through phrase databanks and pronounce phonemes


; Phonemes and other sound codes, used Esteban Molina-Estolano and
;  Matt Reynolds code to get the constants.

; Pauses
_P0         equ 0x00
_P1         equ 0x01
_P2         equ 0x02
_P3         equ 0x03
_P4         equ 0x04
_P5         equ 0x05
_P6         equ 0x06

; Modifiers and control
_FAST       equ 0x07
_SLOW       equ 0x08
_HIGH       equ 0x0E
_LOW        equ 0x0F
_WAIT       equ 0x10
_VOL        equ 0x14
_SPD        equ 0x15
_PTCH       equ 0x16
_BEND       equ 0x17
_PCTR       equ 0x18
_PORT       equ 0x19
_REP        equ 0x1A
_CALL       equ 0x1C
_GOTO       equ 0x1D
_DLY        equ 0x1E
_RST        equ 0x1F

; Phonemes
_IY         equ 0x80
_IH         equ 0x81
_EY         equ 0x82
_EH         equ 0x83
_AY         equ 0x84
_AX         equ 0x85
_UX         equ 0x86
_OH         equ 0x87
_AW         equ 0x88
_OW         equ 0x89
_UH         equ 0x8A
_UW         equ 0x8B
_MM         equ 0x8C
_NE         equ 0x8D
_NO         equ 0x8E
_NGE        equ 0x8F
_NGO        equ 0x90
_LE         equ 0x91
_LO         equ 0x92
_WW         equ 0x93
_RR         equ 0x94
_IYRR       equ 0x95
_EYRR       equ 0x96
_AXRR       equ 0x97
_AWRR       equ 0x98
_OWRR       equ 0x99
```

```
_EYIY       equ 0x9A
_OHIY       equ 0x9B
_OWIY       equ 0x9C
_OHIH       equ 0x9D
_IYEH       equ 0x9E
_EHLL       equ 0x9F
_IYUW       equ 0xA0
_AXUW       equ 0xA1
_IHWW       equ 0xA2
_AYWW       equ 0xA3
_OWWW       equ 0xA4
_JH         equ 0xA5
_VV         equ 0xA6
_ZZ         equ 0xA7
_ZH         equ 0xA8
_DH         equ 0xA9
_BE         equ 0xAA
_BO         equ 0xAB
_EB         equ 0xAC
_OB         equ 0xAD
_DE         equ 0xAE
_DO         equ 0xAF
_ED         equ 0xB0
_OD         equ 0xB1
_GE         equ 0xB2
_GO         equ 0xB3
_EG         equ 0xB4
_OG         equ 0xB5
_CH         equ 0xB6
_HE         equ 0xB7
_HO         equ 0xB8
_WH         equ 0xB9
_FF         equ 0xBA
_SE         equ 0xBB
_SO         equ 0xBC
_SH         equ 0xBD
_TH         equ 0xBE
_TT         equ 0xBF
_TU         equ 0xC0
_TS         equ 0xC1
_KE         equ 0xC2
_KO         equ 0xC3
_EK         equ 0xC4
_OK         equ 0xC5
_PE         equ 0xC6
_PO         equ 0xC7

; Robot sounds
_R0         equ 0xC8
_R1         equ 0xC9
_R2         equ 0xCA
_R3         equ 0xCB
_R4         equ 0xCC
_R5         equ 0xCD
_R6         equ 0xCE
_R7         equ 0xCF
_R8         equ 0xD0
_R9         equ 0xD1

; Alarms
_A0         equ 0xD2
_A1         equ 0xD3
_A2         equ 0xD4
_A3         equ 0xD5
_A4         equ 0xD6
_A5         equ 0xD7
_A6         equ 0xD8
_A7         equ 0xD9
_A8         equ 0xDA
_A9         equ 0xDB

; Beeps
```

```
_B0         equ 0xDC
_B1         equ 0xDD
_B2         equ 0xDE
_B3         equ 0xDF
_B4         equ 0xE0
_B5         equ 0xE1
_B6         equ 0xE2
_B7         equ 0xE3
_B8         equ 0xE4
_B9         equ 0xE5

; Biological sounds
_C0         equ 0xE6
_C1         equ 0xE7
_C2         equ 0xE8
_C3         equ 0xE9
_C4         equ 0xEA
_C5         equ 0xEB
_C6         equ 0xEC
_C7         equ 0xED
_C8         equ 0xEE
_C9         equ 0xEF

; DTMF
_D0         equ 0xF0
_D1         equ 0xF1
_D2         equ 0xF2
_D3         equ 0xF3
_D4         equ 0xF4
_D5         equ 0xF5
_D6         equ 0xF6
_D7         equ 0xF7
_D8         equ 0xF8
_D9         equ 0xF9
_D10        equ 0xFA ; *
_D11        equ 0xFB ; #

; Misc sounds
_M0         equ 0xFC ; Sonar Ping
_M1         equ 0xFD ; Pistol Shot
_M2         equ 0xFE ; WOW

; End of phrase marker
_EOP        equ 0xFF



; Define Constants
COUNT1      res 0x01
COUNT2      res 0x01
TEMP0       res 0x01
TEMP1       res 0x01
ADDR0       res 0x01
ADDR1       res 0x01

; We will store the UART output here as a debugging measure
SENT_DATA   equ 0x80

    org 0x0400
phrase2 ; "Kevin and Raj"
    db  _KE,    _VV,    _IH,    _NE,    _P0,    _P0,    _AY,    _NE
    db  _ED,    _P0,    _P0,    _RR,    _AW,    _JH,    _EOP

    org 0x0000
main

    movlw b'11000111'
    movwf T0CON

    call    uart_init       ; initialize the PIC's UART

    call    speak_phrase1   ; run the speak function
    bra     done


speak_phrase1
    call    sj_stop_voice   ; stop the chip from speaking
    call    sj_clear_buffer ; clear the buffer
    call    sj_start_voice  ; start speaking what is in the buffer

    movlw   0x04            ; set table pointer to 0x0400 (phrase 1)
    movwf   TBLPTRH         ; *see above comment*
    clrf    TBLPTRU
    clrf    TBLPTRL         ; clear the lower part of the pointer
phrase_read_loop
    tblrd*+                 ; read in the phrase data
    movff   TABLAT, TEMP0   ; copy the information where the pointer is to TEMP0 reg.
    movff   TABLAT, POSTINC0

    movf    TEMP0, W        ; send the phoneme to the SpeakJet
    call    uart_put

    movlw   _EOP
    cpfseq  TEMP0           ; continue until the end of phrase marker is reached
    bra     phrase_read_loop

    call    Dlay20
    return


Dlay20                          ; Dealy 20 ms by delaying 5ms four times
    call Dlay5
    call Dlay5
    call Dlay5
    call Dlay5
    return

Dlay5                           ; Dealy 5 ms by dealying 1ms five times
    call Dlay1
    call Dlay1
    call Dlay1
    call Dlay1
    call Dlay1
    return

Dlay1                          ; Delay 1ms (little more than)
;   movlw b'1001111'            ; The Number x 256 to count up to in order to have a 1ms delay
                                ; found by doing (delay_time/clockcycle_time)/256 prescalar for 20
mhz
    movlw b'0000100'            ; for 1mhz clock
    clrf TMR0L                  ; Reset the Timer
Dlay1b
    cpfsgt TMR0L                ; If the timer counter is less than the w-reg, keep going
    bra Dlay1b
  return

; UART subroutines
;  These subroutines deal with the UART port on the PIC, which we are using
;  to communicate with the SpeakJet. See PIC datasheet p 168

uart_init
    movlw   b'00100110'     ; See PIC datasheet p 168
    movwf   TXSTA
    movlw   b'10000000'     ; Turn serial port on, disable receive.
    movwf   RCSTA
    movlw   d'6'            ; Set baud rate to 9600, 6 for 1 Mz and 129 for 20 MHz
    movwf   SPBRG

    bcf     TRISC, TX       ; Set TX to output
    bcf     PIE1, TXIE      ; clear transmit interrupt
    return

uart_put
    btfss   PIR1, TXIF      ; Wait for TXREG to be empty
    goto    uart_put
    movwf   TXREG           ; Transmit byte in W

    movwf   POSTINC1        ; debug thing
    return

; SCP Mode Commands
;  The following subroutines execute SCP Mode commands that allow direct
;  control over ther SpeakJet using the serial connection. See the SpeakJet
;  datasheet p 6 for more information on SCP Mode.

sj_enter_scp                ; '\0' Enter SCP Mode
    movlw   '\\'
    call    uart_put
    movlw   '0'
    call    uart_put
    return

sj_exit_scp                 ; 'X' Exit SCP Mode
    movlw   'X'
    call    uart_put
    return

sj_reset                    ; 'W' Exit SCP Mode and reset SpeakJet
    call    sj_enter_scp
    movlw   'W'
    call    uart_put
    return

sj_verify                   ; 'V' Enunciate "Ready" to verify connection
    call    sj_enter_scp
    movlw   'V'
    call    sj_exit_scp

sj_clear_buffer             ; 'R' Clear Buffer
    call    sj_enter_scp
    movlw   'R'
    call    uart_put
    call    sj_exit_scp
    return

sj_start_voice              ; 'T' Start Enunciating
    call    sj_enter_scp
    movlw   'T'
    call    uart_put
    call    sj_exit_scp
    return

sj_stop_voice               ; 'S' Stop Enunciating
    call    sj_enter_scp
    movlw   'S'
    call    uart_put
    call    sj_exit_scp
    return

done
    bra done
    end
```