ZBasic

Application Note

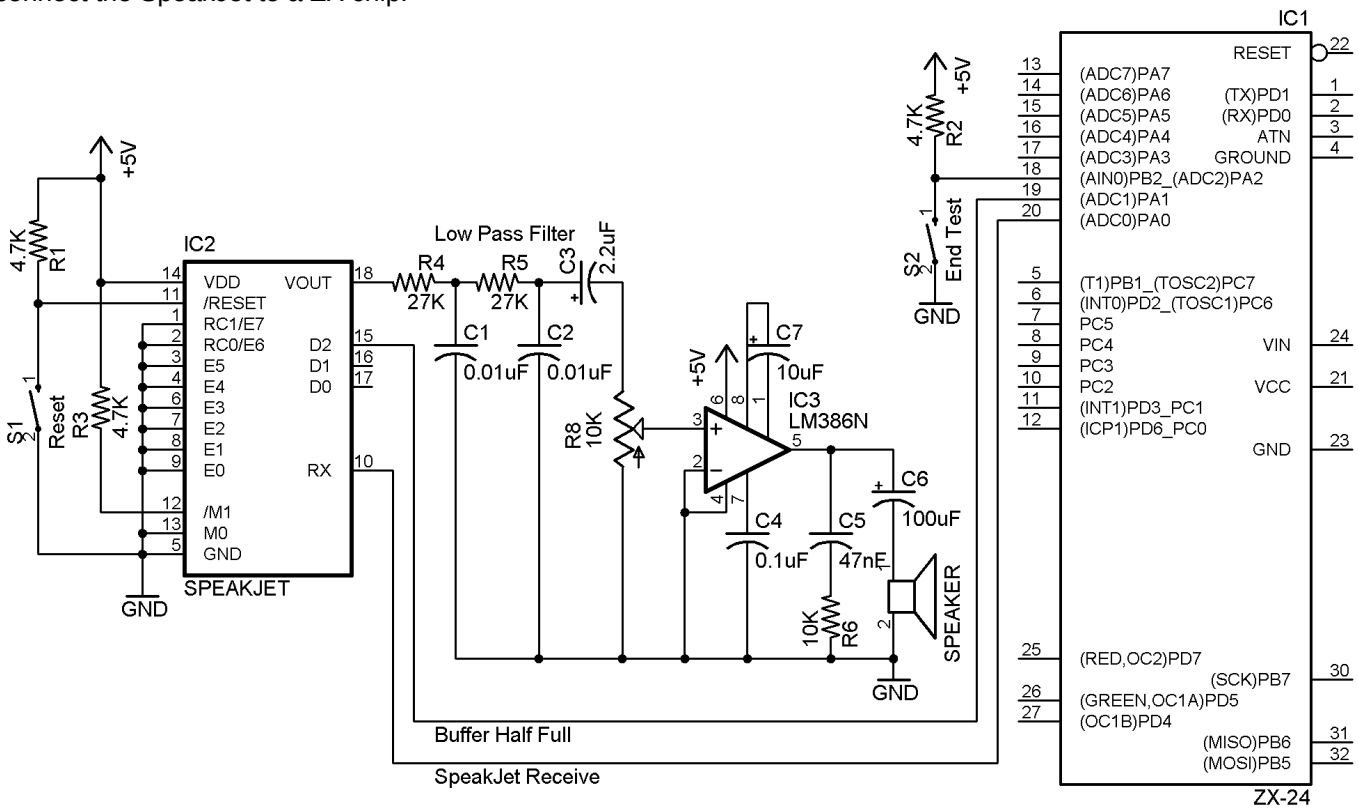
AN-207 Interfacing to SpeakJet Voice and Sound Synthesizer

Introduction

This application note describes how to interface a SpeakJet with a ZX chip and use ZBasic to generate speech or sounds using the SpeakJet chip. The SpeakJet (from Magnevation LLC) is a completely self contained, single chip voice and complex sound synthesizer. The SpeakJet has a built in library of 72 speech elements (allophones), 43 sound effects, and 12 DTMF Touch Tones. The SpeakJet can be controlled by a serial data line from a ZX chip. Other features include an internal 64 byte input buffer, internal programmable EEPROM, three programmable outputs, and direct user access to the internal five channel sound synthesizer. More information can be found at the SpeakJet website (http://www.SpeakJet.com). There are other more capable speech synthesizers in the market but they are much more expensive and are restricted to speech only. The SpeakJet represents a reasonable compromise of voice/sound function and price.

Hardware Hookup

The SpeakJet chip can be connected to a host PC or a ZX chip with very little effort. The simplest connection apart from power can be done with one wire to receive data from a serial connection. The schematic below shows how to connect the SpeakJet to a ZX chip.



Because the SpeakJet is driven from a ZX chip, the event inputs E0-E7 are not used and connected to ground. Although the mode input pins M0 and M1 could be controlled from ZBasic this is unnecessary in most circumstances and therefore M0 and M1 are hardwired to ground and 5V respectively. The output signal from the SpeakJet (V_{OUT}) is connected to a low-pass filter to “round-off” the square-wave-based signal which is then amplified using a standard LM386 circuit and connected to a speaker. The low-pass filter is not strictly necessary but it does result in a less robotic sound.

The ZX-24 (or ZX-40) is connected to the SpeakJet using two wires. The most important connection is the serial transmit connected to receive (pin 10) of the SpeakJet. Optionally the D2 output from the SpeakJet can be

Published December 2005

connected to the ZX chip (pin 19 here) to provide flow control. This line is high when the SpeakJet buffer is more than half full and can be used by ZBasic to control the transmissions to the SpeakJet.

A simple active-low switch is used on pin 18 of the ZX-24 to end the test routine. For brevity the rest of the standard connections to the ZX-24 are omitted. A similar circuit can also be used with a ZX-40.

It is believed that the SpeakJet is based on a PIC 18Fxxxx chip partly because of the placement of +V, ground, V_{OUT} and RX. The Reset line (/RESET) is a software-based reset. The actual PIC reset pin (MCLR) is E3 and although the reset function is turned off, this pin in particular should never be connected to a negative voltage.

Software

This application note also comes with some ZBasic software to drive the SpeakJet. The file SpeakJet.bas is an interface module for the SpeakJet and the file AN207.bas tests the SpeakJet module with some example phrases.

The public interface to the SpeakJet module consists of constants for SpeakJet control codes and three subroutines named InitSpeakJet, TermSpeakJet, and Speak.

The subroutine InitSpeakJet creates a queue and serial port on a given pin for transmitting data to the SpeakJet. InitSpeakJet also sends some startup control messages to the SpeakJet to set the initial volume (100%) and configure the SpeakJet output pins. Here is the essential code for InitSpeakJet:

```
' Needed to save values from initialization
Private cts as Byte
Private sjPort as Byte

' Assume maximum message we will send is 32 bytes (half SpeakJet buffer)
Private Const maxBuffer as Integer = 32

' Queue can be small because Speakjet has a large buffer
Private speakQueue(1 to 10) as Byte

' ************************************************************
' Sub InitSpeakJet
'
' Initialize SpeakJet by setting up serial queue and port
' Parm1 is the serial port number e.g. 3
' Parm2 is the transmit pin to the SpeakJet
' Parm3 is the cts (or flow control pin) input from the SpeakJet
'************************************************************
Public Sub InitSpeakJet(ByVal comPort as Byte, ByVal txPin as Byte, ByVal ctsPin as Byte)
   sjPort = comPort
   cts = ctsPin

   Call OpenQueue(speakQueue, Sizeof(speakQueue))
   Call DefineCom(sjPort, 0, txPin, &H08)
   Call OpenCom(sjPort, 9600, 0, speakQueue)

   Call sendControlCodes(resetSJ.DataAddress, SizeOf(resetSJ))
   Call sendControlCodes(config.DataAddress, SizeOf(config))
   Call sendControlCodes(defaults.DataAddress, SizeOf(defaults))
End Sub
```

The private subroutine sendControlCodes is used by InitSpeakJet and Speak subroutines to copy data out of EEPROM (program memory) and send the bytes to the SpeakJet. . To save valuable RAM the command codes are loaded out of EEPROM into the SpeakJet transmit queue one byte at a time. Because of the slow speed of the connection (9600 baud), this is more than fast enough for the SpeakJet. The public subroutine TermSpeakJet simply closes the serial port (sjPort) specified by InitSpeakJet.

The public subroutine Speak does all of the work to send a byte array of codes to the SpeakJet and can operate in three different modes depending on the setting of the ctsPin (saved from InitSpeakJet subroutine) and the waitNeeded boolean parameter to Speak. The three modes are as follows:

ctsPin	waitNeeded	Description of Mode
0	N/A	No flow control is done and the command codes are put on the SpeakJet transmit queue. If the queue is full for any reason, then the task suspends waiting until all the bytes can be put on the queue.
not 0	true	Flow control is done from the SpeakJet. If the SpeakJet is busy then the Speak subroutine waits until the SpeakJet is not busy (less than half of the buffer is used) to send the command codes to the SpeakJet.
not 0	false	Flow control is done from the SpeakJet. If the SpeakJet is busy then the SpeakJet buffer is flushed and the command codes sent to the SpeakJet. This allows for always the latest output to be presented cutting off any existing output.

The code for the Speak subroutine is shown below.

```
' ************************************************************
' Sub Speak
'
' Send command string to SpeakJet
' Parm 1 is the address in EEPROM of the array of byte to send
' Parm 2 is the length (size) of the array of bytes
' Parm 3 is a boolean to control how to process the buffer half
'      full flow control
'
'************************************************************
Public Sub Speak(ByVal eepromAddress as Long, ByVal length as Integer, _
       ByVal waitNeeded as Boolean)

   ' make sure don't overfill the SpeakJet buffer
   If (length > maxBuffer) Then
       Debug.Print "SpeakJet message too long"
       length = maxBuffer
   End If

   ' perform flow control if the pin is set
   If cts <> 0 Then
       If GetPin(cts) = 1 Then
           ' wait for SpeakJet buffer to be less than half-full
           If waitNeeded Then
               Debug.Print "Wait for SpeakJet"
               Do While GetPin(cts) = 1
                   Call Sleep(10)
               Loop
           ' otherwise simply flush any existing contents
           Else
               Debug.Print "SpeakJet busy so flush buffers"
               Call ClearQueue(speakQueue)
               ' command is executed immediately by SpeakJet
               Call sendControlCodes(flushSJ.DataAddress, SizeOf(flushSJ))
               ' wait for SpeakJet flush to complete
               Call Sleep(10)
           End If
       End If
   End If

   ' send message to SpeakJet
   Call sendControlCodes(eepromAddress, length)
End Sub
```


Published December 2005

Getting Speech from the SpeakJet

The code with this application note concentrates on speech output but it is also possible to use the SpeakJet to generate sounds. Speech is generated by the SpeakJet by sending it command codes that correspond to elements of speech called allophones. For example the short E vowel sound in words such as “met” or “red” is the “EH” allophone which has a SpeakJet code of 131. The PhraseAlator software from Magevation (see the TigerRobotics support page: http://www.tigerbotics.com/support/speakjet.htm) can be used to test out the SpeakJet from a PC and also create spoken words and phrases from allophones. A built-in dictionary of 1400 words provides a good start to creating SpeakJet command codes for speech. An example of the unfiltered output from the SpeakJet for a custom phrase is in zip file associated with this application note.

The codes for common commands, sounds and all the speech elements (allophones) are available as public constants in SpeakJet.bas. Here is an extract of some of the constants:

```
Public Const PA0     as Byte = 0   ' pauses
Public Const PA1     as Byte = 1
Public Const Fast    as Byte = 7
Public Const Slow    as Byte = 8
Public Const Stress  as Byte = 14
Public Const Relax   as Byte = 15
Public Const EY      as Byte = 130
Public Const EH      as Byte = 131
Public Const D0      as Byte = 240 ' DTMF tones
Public Const D1      as Byte = 241
Public Const D2      as Byte = 242
Public Const M0      as Byte = 252 ' sonar pin
```

Phrases can either be sent directly to the SpeakJet from ZBasic or a command can be sent to invoke a phrase preprogrammed in the SpeakJet EEPROM. The example test program (AN207.bas) shows both methods and how to invoke the interface to the SpeakJet module described previously. Here is an extract of this test code:

```
' change this constant to alter the behavior of sending data to the SpeakJet
Private Const doWait as Boolean = false

Private Const serialPort as Byte = 3  ' com3
Private Const tx as Byte      = 20 ' serial port transmit
Private Const cts as Byte     = 19 ' optional for flow control from SpeakJet
Private Const stopPin as Byte = 18 ' just used for testing purpose

' preprogrammed phrase #5
Private countdown as ByteVectorData ({29, 5})
' custom phrase “Welcome to ZBasic”
Private welcome as ByteVectorData ({WW, EH, LE, PA4, KO, AW, MM, PA5, TT, IHWW, PA5,
                         ZZ, IY, PA5, BE, EYIY, SE, IH, Fast, PA4, OK, PA5, EOS})

Public Sub Main()
   ' start test by initializing SpeakJet
   Call InitSpeakJet(serialPort, tx, cts)
   ' main test loop
   Do While GetPin(stopPin) = 1
       Debug.Print "Sending welcome message to SpeakJet"
       Call Speak(welcome.DataAddress,SizeOf(welcome), doWait)
       Debug.Print "Sending countdown message to SpeakJet"
       Call Speak(countdown.DataAddress,SizeOf(countdown), doWait)
       Call Sleep(2.0)
   Loop
   ' stop test by closing down SpeakJet
   Call TermSpeakJet()
End Sub
```

Here is the console output from the above program showing how the SpeakJet is buffer is emptied when it is full and still speaking a different phrase. The audio output gets cut off midstream and the new phrase is started.

Published December 2005

```
ZBasic v1.1
Start of SpeakJet test and don't wait for message to complete
Sending welcome message (23 bytes) to SpeakJet
Sending countdown message (2 bytes) to SpeakJet
SpeakJet busy so reset
Sending welcome message (23 bytes) to SpeakJet
Sending countdown message (2 bytes) to SpeakJet
Sending welcome message (23 bytes) to SpeakJet
Sending countdown message (2 bytes) to SpeakJet
SpeakJet busy so reset
End of SpeakJet test
```

Author

Mike Perks is a professional software engineer who became interested in microcontrollers a few years ago. Mike has written a number of articles, projects and application notes related to ZBasic, BasicX and AVR microcontrollers. Mike is also the owner of Oak Micros which specializes in AVR-based devices including his own ZX-based products. You may contact Mike at mikep@oakmicros.com or visit his website http://oakmicros.com.