Column #78, October 2001 by Jon Williams:

Sounds Like Time for Tricks and Treats

For me, there is no better night of the year than October 31st. Halloween. Better than my birthday (July 25th in case you'd like to send a belated gift...) and Christmas and any other day that is fun. Perhaps I'm a freak. That's okay – Halloween is still my favorite.

Don't get me wrong...I like black clothing but other than October 31st I don't dress any differently any other 39-year-old "normal" guy, and I certainly don't wear make-up unless I'm working on a film or television project. In that case, it's to hide the dark circles, not to accentuate them. And still, I love Halloween.

When I was living in southern California, my friend Paul and I used to create these elaborate, crawl-through haunted houses out of commercial cardboard shipping boxes. These boxes were very strong. The last time I was involved we actually made a two-story model. Inside, we would paint it black to make it even darker, we would have dangling threads, sound effects, pressure switches that set off flashbulbs and even a couple of holes where we could reach our hands through for an extra fright. Again, these things were designed for kids – and kids seem to come in all ages. We didn't have to worry about dissembling that haunted house; the adults crawling through it did a good job.

When I moved to Texas I really missed that. Then my friend Kelley moved into the city and asked me to decorate her lawn for Halloween. What I discovered is that there are a lot of dedicated Halloween prop and decoration builders that are doing some really amazing things (and more than a few are using BASIC Stamps). We created a cool display with headstones and a decapitated witch and spiders and snakes and all the usual Halloween stuff. Her lawn was the hit of the neighborhood.

Our success just got me going and I ended up buying several books on prop building. As you read this, I'm in full-swing production on several new props that will decorate the yard of my acting coach. She throws a big party each Halloween, figuring actors will really get into it. Of course, we do.

A lot of the stuff I'm building is mechanical in nature and certainly not appropriate here. You can be sure though, many of my decorations are being controlled by BASIC Stamps. In the end, there will be five or six Stamps running everything that moves, has lights or makes sounds.

Since there's not a lot of time between now and Halloween, we're going to do an incredibly easy project; no tricks, just a treat. One that is Halloween-cool and can be assembled on a solderless-breadboard (like a BOE) in about two minutes. And it gives me the opportunity to introduce a couple of new, Stamp-friendly products: the key fob transmitter/receiver pair from RF Digital and the new QV306M1 sound module from Quadravox.

Sounds From Beyond The Grave

This little project is what I needed for our graveyard section. I wanted sounds to come from it, but not at preprogrammed intervals or in any particular order. What I needed was some sort of remote control. That way, if someone gets too close, I can play a "Get out!" sound file and make them think twice before proceeding.

I could use a remote switch, but that would mean stringing wires and would limit my mobility. I could use IR like we did a couple months ago ("Control From The Couch" – August issue), but that would mean that I would have to be in sight of the IR receiver. Then a solution came: RF Digital came out with the key fob transmitter/receiver pair.

The key fob transmitter looks just like the transmitter that you probably have to unlock your car or set/disable its alarm. It has five buttons: one in the center with four, slightly larger buttons that surrounding it. The buttons are comfortably spaced so there's no problem with holding the transmitter in one hand and operating any of the buttons with your thumb.

Figure 78.1: Tricks and Treats Circuit
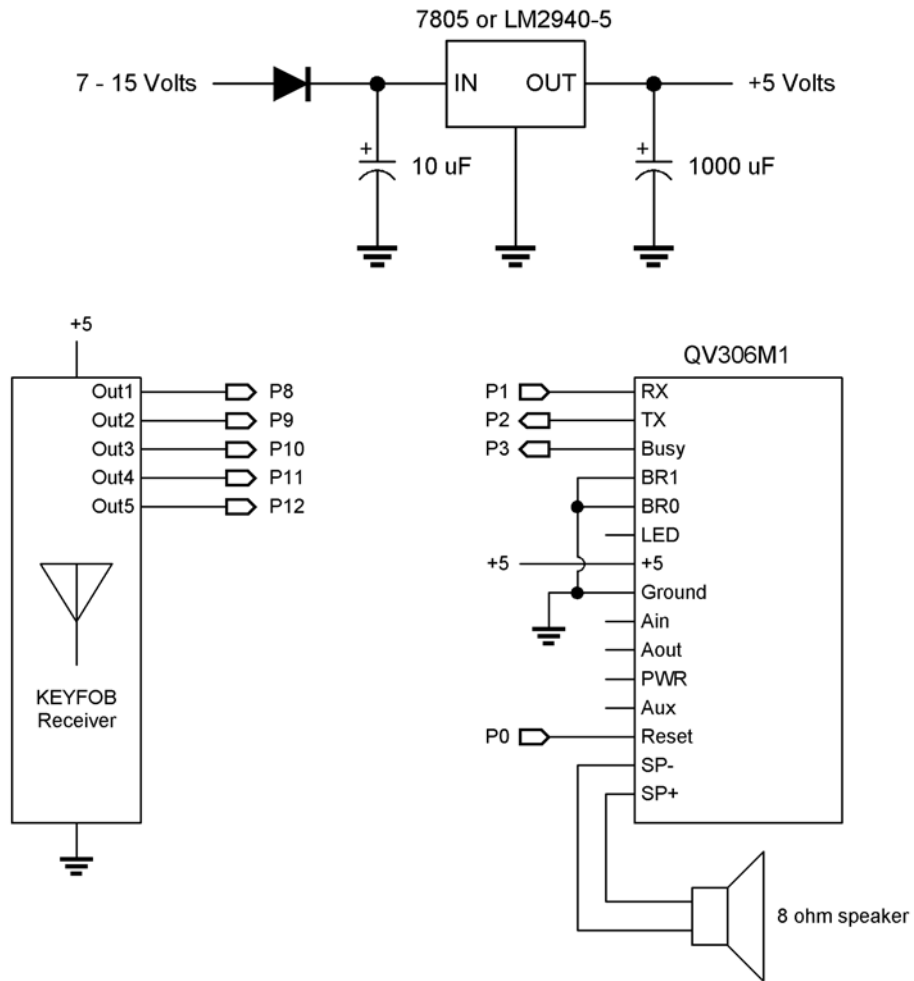
The key fob transmitter/receiver pair comes is available in two frequencies: 418 MHz and 433 MHz. The range is about 75 feet – outdoors without obstructions. This is perfect for me since I won't be more than 20 or 30 feet from the yard at any time.

The key fob receiver is a small (0.75" x 2.5"), single-sided PCB with a simple wire antenna and a seven-pin SIP header. The pins make it perfect to plug into a solderless breadboard. All we have to do is provide regulated five volts and ground, then press a button on the key fob transmitter. When we do, the associated output pin will go high. It's that easy.

Bigger Sound From A Smaller Package

Back in June and July I showed you how to use the Quadravox QV306M4 to add sound to your BASIC Stamp projects. Well, my friends at Quadravox have been working hard and listening to customer feedback about the product. They've recently introduced a new module, the QV306M1.

The M1 module is mechanically, electrically and software compatible with the M4. The only change you would have to make is in the volume setting. M4 volume level is from 0 to 31, while the M1 volume level is from 0 to 63.

The M1 is considerably smaller than the M4. It's just over one-and-a-half inches wide and about three-quarters of an inch tall. This was accomplished by using surface mount components. This being the case, there's not much filtering on the power supply input so you really need to do that externally.

There are two more important features that make the M1 very attractive: it has a better amplifier, capable of delivering a full watt of sound (the M4 is 300 mW) and it has a serial buffer that lets us assemble a "string" of files and then play that string with no discernable gaps in between. This last feature is really great and has been the biggest annoyance among M4 users.

Let's be clear though, it's not a flaw with the M4. The ISD chip that is used on the module makes extra noises (clicks and pops) when accessed in sequential mode. The guys at Quadravox know this (they spend most of their time developing ISD development tools and they definitely have a handle on this), so they have incorporated logic into the M4 controller that mutes the amplifier between sound files. This takes a bit of time. By design, the M1 does the same thing, it's just that a better amplifier is available so the process is significantly less noticeable.

Another reason for delays between speech/sound files is the external processing (from the Stamp, for example). This problem is solved with three new commands in the M1:

$F9   Open string buffer
$FA   Close string buffer
$FB   Play string buffer

When the $F9 command is sent, sounds will not immediately play (like in direct mode, $F0). Instead, they'll be queued in a buffer that can be up to 48 bytes long. When we've finished putting sound file numbers into the buffer we send $FA. This tells the M1 that we're done building the buffer. Sending $FB plays it; cleanly, without gaps – it's a beautiful thing. And, just as a bonus, if we want to replay the buffer we can without rebuilding it.

Playing Spooky Sounds

Okay, let's build the project. The schematic in Figure 78.1 shows just how simple this project is. I kid you not: I assembled the hardware in about two minutes on a Parallax BOE (Board of Education). Of course, you can use any solderless breadboard – just be sure that you provide a clean, five-volt source to the key fob receiver and to the QV306 (see Figure 78.2 for an example).

The connections between the key fob receiver and the Stamp are direct connections. The receiver provides a digital high output when the corresponding button is pressed (yes, we can press more than one button at the same time), otherwise the outputs are low. We'll configure pins 8 through 12 as inputs and watch them for a button press.

The connections between the Stamp and the QV306M1 are "maxed out" – so to speak. In many applications we can get away with fewer. Please note that by using these connections, the QV306M4 can be used with this project as well. A difference between the M1 and M4 is that the M4 requires the Reset line to bring it out of sleep mode. Any serial input to the M1 when it is asleep will cause it to wake (the command will be ignored). And with the ability to build and play strings, we really don't need the Busy line when using the M1.

Let's chat a bit about the program's design before we get into code details. What I did is collect a bunch a Halloween-oriented WAV files from the Internet, process them to meet the requirements of the QV306 and downloaded them to the QV306 using the QV430 programming cradle. Remember, you can "roll your own" programmer using a BASIC Stamp. Download the July column from the Parallax web site for details on the DIY programmer and on processing WAV files for the QV306.

The software is really just as simple as the circuit. About the trickiest thing we'll do here is make it compatible with the QV306M4 and the QV306M1 by using the device-type request function. Keep in mind that the M1 variant has the string play function and we'll take advantage of it.

Notice that in the connections definitions we're renaming (aliasing) the InH register (inputs for pins 8 through 15) as *KeyFob*. This will let us grab all five receiver bits at once. This leads us to an important consideration: since we're only using five bits, it's a good idea to mask out the unused bits. That way, the unpredictable results retuned by the floating inputs will not interfere

with proper operation program of the program. We could use pull-down resistors on the unused inputs, but this just adds extra parts.

After defining our variables, we'll store the file numbers for sound strings in **DATA** statements. You'll notice that each string is terminated by $FF. The terminating byte allows us to keep the string length variable. $FF is used for convenience. Technically, any value greater than $EF (239; the highest possible QV306 sound file) will work with the program.

The QV306 is initialized by taking its Reset line low for 100 milliseconds, then bringing it back high. After the device is reset we pause for two seconds to allow the QV306 to take care of internal housekeeping.

Since this code is designed to work with either the QV306M4 or the newer QV306M1, the get device-type command ($FF) is sent to the sound module. The return should be 54 for the M4 module or 55 for the M1 module. **LOOKDOWN** is used to set the variable *devType* to 0 or 1 based on the detected device.

You might wonder why we used **LOOKDOWN** instead of a simple subtraction to set the *devType* variable. The reason is this: if the module doesn't respond (perhaps the QV306 TX line is not connected), we will assume it's an M4 module and treat it accordingly. Since the M1 uses a superset of the M4 commands, this will allow either module to work fine. If it is, indeed, an M1 module, the sound may be a little low.

Once we know the device type we'll set its volume. Refer to the July article for using a potentiometer (read with **RCTIME**) to set the volume if you need it to be variable. I selected default volume levels for each module type base on the speaker I'm driving. This is the one area you'll definitely need to adjust for your setup.

Finally, we'll initialize the value of *lastKey*. It's important to initialize *lastKey* to something different than *theKey* so that the string will be built in the QV306M1 if Button1 is pressed first.

In the main loop of code, the first thing that happens is that the inputs from the key fob receiver are scanned. The **NCD** function is used to return the button number pressed. This means we'll only look at one button at a time. If you want to allow multiple button presses, you should add a bit of debounce routine (and drop **NCD**). Debouncing isn't necessary for this program because that is handled by the key fob transmitter and receiver. Note that the constant called KeyMask is used to mask out the unused bits.

The result of **NCD** will be between zero (no press) and five. If zero, we simply loop back and look again. For a non-zero values, we'll subtract one and use **LOOKUP** to load the first byte of our stored sound file string.

Next, we look at the device type. If it's an M4, we can start sending (playing) chunks from our sounds string. If the device is an M1, we look to see if the requested string is the same as the last one. If yes, we'll play it. If not, we'll send the Open String command to the M1.

The next section (at Get_Sound_File) loops through the EEPROM reading the sound file numbers and sending them to the QV306. We need to monitor the busy line during this process, especially if the module is an M4 and the sound will be playing. When $FF (end of string) is encountered the string is complete. If the module is an M1, we'll close the string then send the Play String command.

At the end, we'll make sure that there's ot button pressed before moving on. This will prevent unintentional repeat plays of the same string.

Go To It

You see, this is an easy project so get to it. There are any number of modifications you could make to ad you your own input devices. Feel free to download the Halloween WAV files I used from Nuts & Volts or from the Parallax web site – or go get your own; there's a bunch of them available on the Internet. And for those of you who might be new to the QV306 sound module, be sure to review the June and July issues for additional details.

Column #78: Sounds Like Time for Tricks and Treats

```
' -----[ Title ]------------------------------------------------------------
' Program Listing 78.1
' File...... HALLOWEEN.BS2
' Purpose... Wireless Halloween Sound FX Controller
' Author.... Jon Williams
' E-mail.... jonwms@aol.com
' Started... 09 SEP 2001
' Updated... 11 SEP 2001

' {$STAMP BS2}


' -----[ Program Description ]-----------------------------------------------
'
' This program monitors pins 8 - 12 for input from a Parallax keyfob receiver.
' Based on the inputs, it plays sounds stored in a QV306M4 or QV306M1 sound
' module.

' QV306M4 / QV306M1 Connections:
'
'  1 (RxD)              Stamp.P1
'  2 (TxD)              Stamp.P2
'  3 (Busy)             Stamp.P3
'  4 (BR1)              Ground
'  5 (BR0)              Ground
'  7 (+5)               +5 volts
'  8 (Gnd)              Ground
' 14 (Reset)            Stamp.P0
' 15 (Sp-)              8 ohm speaker -
' 16 (Sp+)              8 ohm speaker +
'
' Keyfob Receiever Connections:
'
'  1 (Gnd)              Ground
'  2 (+5)               +5 volts
'  3 (Out1)             Stamp.P8
'  4 (Out2)             Stamp.P9
'  5 (Out3)             Stamp.P10
'  6 (Out4)             Stamp.P11
'  7 (Out5)             Stamp.P12


' -----[ Revision History ]--------------------------------------------------
'


' -----[ I/O Definitions ]---------------------------------------------------
'
KeyFob         VAR    InH                      ' from keyfob receiver
```

```
QV_RST          CON     0                       ' to QV306M4 or QV306M1
QV RX           CON     1
QV TX           CON     2
QV BUSY         VAR     In3



' -----[ Constants ]-------------------------------------------------------
'
T2400           CON     396                     ' 2400 baud, true
IsBusy          CON     0

' Quadravox Commands

QV Play         CON     $F0                     ' play direct file
QV Stop         CON     $F6
QV Sleep        CON     $F8
QV OpenStr      CON     $F9                     ' open string buffer (QV306M1)
QV_CloseStr     CON     $FA                     ' close string buffer (QV306M1)
QV PlayStr      CON     $FB                     ' play string buffer (QV306M1)
QV Volume       CON     $FC
QV Reset        CON     $FD                     ' software reset
QV Rev          CON     $FE                     ' module revision
QV Type         CON     $FF                     ' module type

QV306M4         CON     0
QV306M1         CON     1


KeyMask         CON     %00011111               ' mask for pin inputs



' -----[ Sound Files ]-----------------------------------------------------
'
 attack         CON     0                       ' animal attack.wav
 belch          CON     1                       ' belch.wav
 boing          CON     2                       ' boing.wav
_bonk           CON     3                       ' bonk.wav
_door           CON     4                       ' creaking door.wav
 dingdong       CON     5                       ' ding-dong.wav
 zap            CON     6                       ' electric current.wav
 laugh1         CON     7                       ' evil laugh 1.wav
 laugh2         CON     8                       ' evil laugh 2.wav
_explode        CON     9                       ' explode.wav
_barf           CON     10                      ' funny barf.wav
 getout         CON     11                      ' get out.wav
 glass          CON     12                      ' glas.wav
 gunshot        CON     13                      ' gunshot.wav
 hh             CON     14                      ' happy halloween.wav
_moan1          CON     15                      ' moan 1
_moan2          CON     16                      ' moan 2
 phone          CON     17                      ' phone ring.wav
_scream         CON     18                      ' scream.wav
```

```
_shotgun        CON     19                      ' shotgun.wav
 thunder        CON     20                      ' thunder.wav
 werewolf       CON     21                      ' werewolf.wav


' -----[ Variables ]-------------------------------------------------------
'
qvData          VAR     Byte                    ' data to/from QV306
devType         VAR     Bit                     ' device type
theKey          VAR     Nib                     ' current key press
lastKey         VAR     Nib                     ' last key press
addr            VAR     Word                    ' ee address of sound file


' -----[ EEPROM Data ]-----------------------------------------------------
'
Btn1            DATA     hh,  laugh1, $FF
Btn2            DATA    _scream, _gunshot, _bonk, _glass, $FF
Btn3            DATA    _dingdong, _door, _attack, $FF
Btn4            DATA     belch, laugh2,  barf, $FF
Btn5            DATA     getout, moan1, werewolf, thunder, $FF


' -----[ Initialization ]--------------------------------------------------
'
Initialize:
  LOW QV RST                                    ' Reset the QV306
  PAUSE 100
  HIGH QV RST
  PAUSE 2000

  GOSUB QV GetType                              ' get device type from module
  LOOKDOWN qvData, [54, 55], devType            ' 0 = QV306M4, 1 = QV306M1

  DEBUG CLS, "Device is QV306M", "4" - (devType * 3)

  qvData = 25                                   ' default volume for QV306M4
  IF (devType = QV306M4) THEN Set Volume
  qvData = 50                                   ' defualt volume for QV306M1

Set Volume:
  GOSUB QV_SetVolume

  lastKey = ~theKey                             ' force string build first time


' -----[ Main Code ]-------------------------------------------------------
'
Main:
  theKey = NCD (KeyFob & KeyMask)               ' get receiver inputs
  IF (theKey = 0) THEN Main                     ' button pressed?
```

```
      LOOKUP theKey-1, [Btn1, Btn2, Btn3, Btn4, Btn5], addr

      IF (devType = QV306M4) THEN Get Sound File  ' skip this if 306M4
      IF (theKey = lastKey) THEN Play M1 String
      SEROUT QV_RX,T2400,[QV_OpenStr]             ' open buffer on 306M1

Get_Sound_File:
      READ addr, qvData                          ' get sound file
      IF (qvData > 239) THEN Done

QV306_Busy:
      IF (QV_Busy = IsBusy) THEN QV306_Busy       ' wait for Busy to release
      SEROUT QV_RX,T2400,[qvData]                 ' send the phrase
      addr = addr + 1                             ' point to next phrase
      GOTO Get_Sound_File                         ' send it

Done:
      IF (devType = QV306M4) THEN Do_Again        ' if 306M4 - we're done
      SEROUT QV_RX,T2400,[QV_CloseStr]            ' ..otherwise, close buffer

Play M1 String:
      SEROUT QV_RX,T2400,[QV_PlayStr]             ' play it
      lastKey = theKey

Do_Again:
      qvData = KeyFob & KeyMask                   ' check inputs
      IF (qvData > 0) THEN Do_Again               ' make sure they're clear
      GOTO Main                                   ' get next button
      END


' -----[ Subroutines ]-------------------------------------------------------
'
QV GetType:
      qvData = 0
      SEROUT QV_RX,T2400,[QV_Type]                ' send request
      SERIN  QV_TX,T2400,2000,NoType,[qvData]     ' get type
NoType:
      RETURN


QV_SetVolume:
      IF (QV_Busy = IsBusy) THEN QV_SetVolume     ' wait for Busy to release
      SEROUT QV_RX,T2400,[QV_Volume]              ' send volume command
      SEROUT QV_RX,T2400,[qvData]                 ' send volume level
      RETURN
```