Speaking Volt Meter

by Ben Rowland, July 2010

Abstract

ARM devices are very popular today as they combine a large memory pool with a high speed processing unit. The ARM7 devices in particular are well suited to an embedded environment. In this article Ben utilises the large memory of the ARM device and creates a speaking volt meter using stored phonemes strung together to create words.

Requirements

Software:

- Professional licence of Flowcode v4 for ARM

Hardware:

- EB016 Patch board
- Small generic speaker

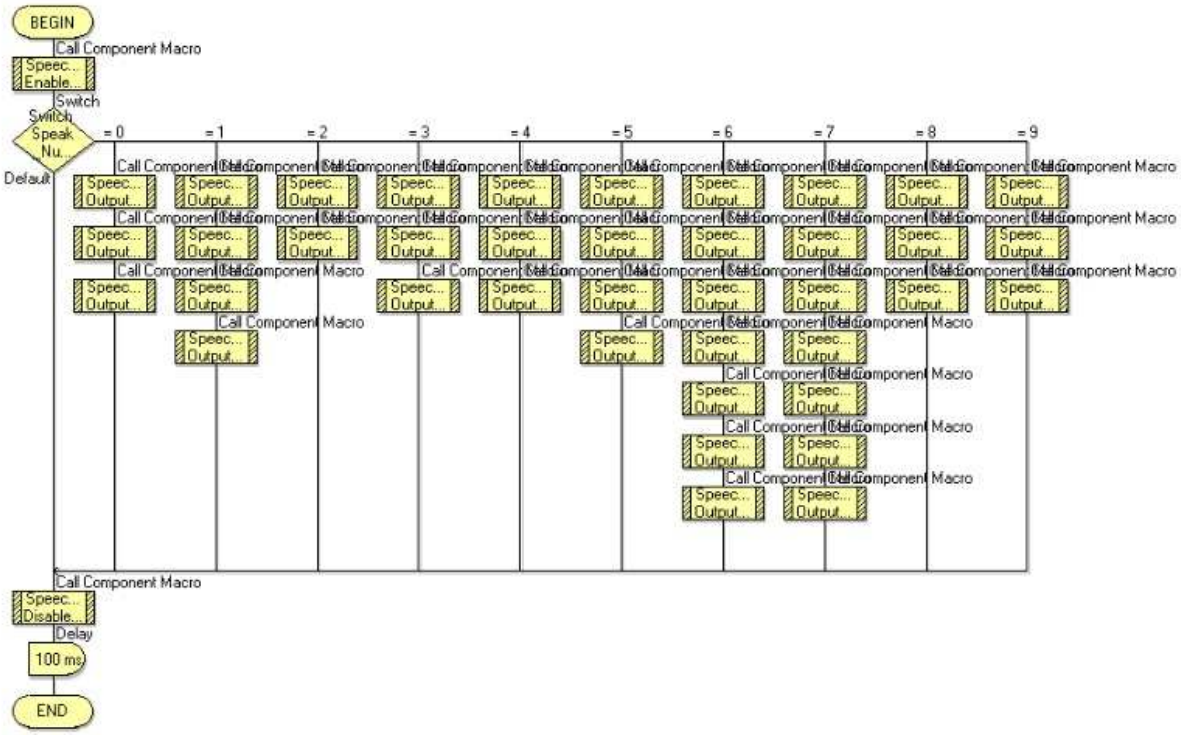
Speaking Volt Meter

ARM devices are very popular today as they combine a high speed processing unit with a large amount of memory. While the ARM9 devices are more suited to a Linux environment the ARM7 devices are much more suited to the embedded environment. A good application of the ARM is the Flowcode Speech component which uses phonemes to build up words and phrases. In this months article we have put together a simple program to demonstrate the speech capability on the ARM devices. As the ARM comes with plenty of available memory it is possible to load the entire phoneme set from the speech component and therefore be able to create any combination of phonemes to in theory build up any word or phrase.

To start the program off the first thing that was created was a macro called Speak_Number to output a number as speech data. The number is passed to the macro as a parameter and then fed into a switch case where the appropriate number is found and output. Each numeric word was made by stringing together two or more phonemes. The phonemes chosen were taken from a library of phoneme words but the actual phonemes used can be edited to your tastes.

Here are the phonemes used to create the numbers,

0: “ZZ” “YR” “OW”
1: “WW” “SX” “AA” “NN1”
2: “TT2” “UW2”
3: “TH” “RR1” “IY”
4: “FF” “FF” “OR”
5: “FF” “FF” “AY” “VV”
6: “SS” “SS” “IH” “IH” “PA3” “KK2” “SS”
7: “SS” “SS” “EH” “EH” “VV” “IH” “NN1”
8: “EY” “PA3” “TT2”
9: “NN1” “AY” “NN1”



Flowcode Diagram

Once the number speaking routine was created the next thing that was needed were routines to output the words “point” and “volts”. As these words did not exist in the phoneme library they had to be made from scratch. Here is the list of phonemes that were used to build up the words. By trial and error you can build up the phonemes and then use the “PAx” to create extra delays to add subtle changes to the way the phonemes sound in the word.

Point: “PP” “OY” “NN1” “TT1” “PA1” “TT1”

Volts: “VV” “OW” “PA2” “LL” “TT1” “TT1” “SS”

Now we have a library of all the words and numbers we need to create our speaking volt meter. The next step is simple. We add an analogue component and a switch component to the program. The switch component is used to hold the program loop until we are ready to take a voltage reading. Once the switch has been pressed the analogue voltage is read using the read as float method. We extract the integer portion of the floating point number by simple copying the float number to a byte variable. Note that if you are using a PIC then you will have to use the conversion function available from the calculation icon properties. Extracting the real portion of the number uses a similar approach. First we subtract the integer portion of the number from the original float variable. Next we multiply the float variable by 100 to allow two of the fraction numbers to move across into the integer portion of the float variable. We then do the same process of saving into a byte variable to strip off all the data after the decimal point.

For Example:

Float variable = 2.46256

Integer Byte = 2

Float variable = 2.46256 – 2 = 0. 46256 * 100 = 46.256

Real Byte = 46

We then use a method of division and modulus to strip out each digit in turn and pass the numeric value to the Speak_Number function.

For more background information regarding the phonemes and speech there is a datasheet for an old IC that used to use this phoneme approach on a BBC micro. The datasheet can be found by doing a Google search for “SP0256”.

Further reading

Below are some links to other resources and articles on related subjects, and technical documentation relating to the hardware used for this project...

Flowcode:	http://www.matrixmultimedia.com/flowcode.php
Eblocks:	http://www.matrixmultimedia.com/eblocks.php
Learning Centre:	http://www.matrixmultimedia.com/lc_index.php
User Forums:	http://www.matrixmultimedia.com/mmforums
Product Support:	http://www.matrixmultimedia.com/sup_menu.php