Embedded C Routines for the Micromite

MMBasic for the Micromite allows a BASIC program to load a machine code module and execute the code in that module. Because the module is written in the C language it can run much faster than a BASIC program and can more easily access the special hardware features of the PIC32 microcontroller.

However, it does require that the programmer has experience with programming in C (not an easy subject to learn) and has a good working knowledge of MPLAB X and programming for the PIC32. This guide provides an outline of how to write an embedded C module but it does not explain how to write C programs. For a good introduction to C see: http://microchip.wikidot.com/tls2101:start

CFunctions and CSubs

Two types of embedded modules are supported. These are a CSub which is a subroutine (ie, it is used as a command) and a CFunction which is a function (ie, it can be used in expressions and returns a value). Both are similar and will be referred to in this tutorial as "embedded C routines".

Using an Embedded C Routine

To use an embedded C routine you must insert the definition of the routine somewhere in your BASIC program. The exact spot is not important but at the end of the program is typical. Having done that you can refer to the CSub or CFunction as a normal subroutine or function.

For example:

```
SerialTx 2, 19200, "Hello World"
CSub SerialTx Integer, Integer, String
    00000008
    00001021 40824800 40024800 0044102B 1440FFFD 00000000 03E00008 00000000
    27BDFFC8 AFB40020 3C149D00 8E820000 AFB00010 8C500000 8CA30000 00108042
    0203001B 006001F4 AFB10014 8E820010 00808821 8C840000 AFBF0034 AFB60028
    AFB3001C AFB20018 00C0B021 24050008 00003021 AFBE0030 AFB7002C AFB50024
    0040F809 00008012 8E240000 8E820014 0040F809 24050001 8E240000 8E820024
    0040F809 24050006 00409021 8E240000 8E820024 0040F809 24050005 00409821
    8E240000 8E820028 0040F809 2610FFFB 82C30000 18600027 24110001 00518804
    24150001 AE710000 02002021 0411FFC4 00000000 241E0008 24170001 10000009
    02D5A021 AE510000 02002021 0017B840 27DEFFFF 0411FFBA 00000000 13C0000D
    32F700FF 82820000 02E21024 1440FFF5 00000000 AE710000 02002021 0017B840
    27DEFFFF 0411FFAE 00000000 17C0FFF5 32F700FF AE510000 02002021 0411FFA8
    00000000 82C20000 02A2102A 1440FFDD 26B50001 8FBF0034 00001021 00001821
    8FBE0030 8FB7002C 8FB60028 8FB50024 8FB40020 8FB3001C 8FB20018 8FB10014
    8FB00010 03E00008 27BD0038
End CSub
```

When run, this program will send the string "Hello World" as a serial stream out of the Micromite's pin 2 at 19200 baud. Note that this embedded C routine is the same as SerialTx which is included and described in the normal Micromite firmware distribution.

More Details

An embedded routine in MMBasic must start with the keyword "CSub" or "CFunction" followed by its name and a sequence of 8-digit hex words. It is terminated by an "End CSub" or "End CFunction" keyword. The name is used to identify the routine to BASIC programs and must follow the rules for a variable name in MMBasic.

The data within the routine is a sequence of 8-digit hex words. Each word must be separated by one or more spaces or a new line.

The first word is the offset (in 32 bit words) to the entry point for the embedded routine. The above CSub is a good example of this, the first word contains the number 8 because the main routine starts eight words into the routine (the linker often places smaller functions before the main routine).

Each of the following data words represents each 32-bit instruction in MIPS machine code. The number of instructions that can be included is limited only by the amount of available memory. The routine is terminated by an "End CSub" or "End CFunction" keyword.

When a BASIC program is saved to flash MMBasic will search through it looking for any CFunction or CSub commands. The machine code specified will then be extracted and programmed into flash memory. This code becomes part of MMBasic and will be called whenever the specified CFunction or CSub name is encountered (it is erased when a new program is saved).

Multiple CFunction and CSub commands can be used if multiple embedded C routines are required. During execution MMBasic will skip over the CFunction and CSub commands and the data specified. This means they can be placed anywhere in the program.

Typed Parameters

A feature that is new in version V5.2 and later of MMBasic is the ability to specify the type of the parameters passed to a C routine and the value returned from a CFunction. This is done by specifying the types on the command line of the C routine's definition as follows:

```
CSub CSubName type1 [, type2 [,…]]
```

or

```
CFunction CFunName( type1 [, type2 [,…]] ) typer
```

The types are a comma separated list of FLOAT, INTEGER or STRING. The returned value of a CFunction ('typer') is also one of these keywords.

For example:

```
CSub SMid FLOAT, FLOAT, INTEGER, INTEGER
    … data …
End Csub

CFunction SGetStr(FLOAT, INTEGER, INTEGER) STRING
    … data …
End CFunction

CFunction SLen(FLOAT) INTEGER
    … data …
End CFunction
```

MMBasic will use these types to either issue an error message if the wrong type of variable is supplied as a parameter or to convert a parameter if it can. For example, if a parameter is specified as an integer and a float is provided MMBasic will convert the float to an integer before passing its address to the C routine. This make it much easier for the C routine writer as they can be sure that the parameter passed is an integer and so complicated checking for an accidental float parameter is not required.

Double Precision Floating Point

The Micromite Plus now uses double precision floating point (the standard 28 and 44-pin Micromites still use single precision). Embedded C routines that manipulate single precision floating point should run unchanged on the Micromite Plus as MMBasic will convert any floating point parameters to single precision before passing them to the embedded routine and will convert any returned float values back to double precision. Also, the float functions provided by MMBasic for use by embedded routines still use single precision.

The only exception is when a floating point array is passed as an argument. Such an array will not be converted (it will remain an array of doubles) so any embedded routine that expects an array of

single precision floats will not work. On the other hand, this is a way of passing and returning double precision floating point data to a routine designed to work with doubles.

Using the Library Feature

Some embedded C routines can get very large and as a result a lot of program memory can be wasted by holding the hex codes for the routine in the program (this is needed so the whole program can be edited using the built in MMBasic editor). It is also tedious inserting the code into each program that may run.

Using the LIBRARY feature it is possible to add embedded C routines (and normal BASIC code) to MMBasic and make them permanent and part of the language. Routines saved in the library area will not show when LIST is used and will not be deleted when a new program is loaded or NEW is used. However they can be called from within the main program and can even be run at the command prompt (just like a built in command or function).

When the routines are transferred to the library space MMBasic will compress them by removing comments, extra spaces, blank lines and the hex codes in CSubs and CFunctions. This recovers a lot of program memory when used on large embedded C routines.

Refer to the "Micromite User Manual" for more details.

Creating Embedded C Routines

Overview

An embedded C routine can accept up to ten arguments and, in the case of a CFunction, returns an integer, string or float. All arguments passed to the embedded routine are pointers to the memory allocated to a variable or the result of an expression. They can be pointers to integers, strings or an array of integers or strings.

Floating point constants and variables can also be passed but they need to be handled carefully within the embedded C routine as the C compiler will generate invalid code for normal floating point operations. Peter Mather's tutorials (referenced below) provides more detail.

There are a few points to note:

- A CFunction normally returns a value which is a 64-bit integer (a long long int in C) however this can be changed (see Typed Parameters later). A CSub does not return a value and MMBasic will ignore it if it is provided.
- Because the arguments are pointers to the memory allocated to the variables in MMBasic your C program can modify this memory and this is another way of returning data to MMBasic. If you pass an expression the argument will be a pointer to the result of the expression (integer, float or string) but changing it will not achieve anything.
- Integers are 64-bit signed numbers (called long long int in C) and occupy 8 bytes. Because the MIPS processor is little endian the pointer actually contains the address of the least significant byte. This means that you can declare the argument as a pointer to a char (8-bits), a short int (16-bits) or int (32-bits) and it will still work as long as the contents of the variable do not exceed these sizes.
- Strings are passed as a pointer to the first byte of the memory allocated to a string (which defaults to 256 bytes). In MMBasic the first byte is the length of the string (in characters) and the 2nd and subsequent bytes are the characters in the string.
- Floats can also be passed (as a pointer) but, as pointed out above, you need to me careful manipulating them because the compiler will embed calls to C library functions which cannot be accessed from within an MMBasic C routine.

- Arrays are passed by using the array name with empty brackets (ie, with no dimensions). For example: `x = CFun(a%, b%(), c%)`. In the embedded C routine the argument will be a pointer to the first element of the array.

An embedded C routine has some important restrictions and issues:

- It cannot call any library functions (eg, toupper(), strcmp(), etc) and it cannot do anything that will cause the compiler to call a library function (eg, manipulate a float). Note that you can call functions defined within the C program module.
- It cannot define data that is const (ie, data that will be stored in flash memory) or static (ie, variables defined outside of a function). This means that you cannot use something like "const int var" or "static int var" and you cannot create literal strings (ie, "foo"). There are ways around this restriction (see Peter Mather's tutorials referenced below).
- It is hard to debug embedded C routines and most programming errors will cause a MIPS processor exception which in turn will force an immediate reboot of the processor.
- MMBasic requires that the embedded routine contain only position independent code (see below for how to specify this) but regardless of the compiler's settings the compiler can sometimes ignore these settings. If this happens the embedded routine will cause a MIPS processor exception and force an immediate reboot of the processor.

Tutorials

Peter Mather (matherp on the Back Shed forum) has produced an excellent series of tutorials which describe how to write CFunctions by example. They are essential reading for anyone who wishes to create embedded routines. They can be accessed by following links:

http://www.thebackshed.com/forum/forum_posts.asp?TID=7965
http://www.thebackshed.com/forum/forum_posts.asp?TID=7973
http://www.thebackshed.com/forum/forum_posts.asp?TID=8038
and http://www.thebackshed.com/forum/forum_posts.asp?TID=8389

Another good resource is the embedded C routines included with the Micromite firmware (http://geoffg.net/micromite.html#Downloads). Some routines include the C source code and it is recommended that you experiment by recompiling and modifying some these before writing your own routines. That way the compiler and tools can be verified as working correctly.

Accessing MMBasic Functions

MMBasic contains a table of handy functions that can be accessed from within an embedded C routine. These functions include getting and sending characters from and to the console, functions to manipulate I/O pins, functions to draw on an attached LCD panel, etc.

To access these the file CFunction.h should be included at the start of your C program. Eg:

```
#include "CFunction.h"
```

You can then use the functions that it defines in your embedded C routine as you would use a normal function. At this time there is little documentation on the functions defined in CFunction.h so you should request a copy of the source code for MMBasic (from http://mmbasic.com) and use that as your reference. The source to SerialTx and SerialRx also provide examples.