The Micromite CFunction

The Micromite CFunction feature allows a BASIC program to load a machine code module into MMBasic and execute the code in that module. Because the module is written in C or MIPS assembler 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 or assembler and has a good working knowledge of MPLAB X and programming for the PIC32. This guide provides an outline of how to write a CFunction module but it does not explain how to write C programs.

Writing a CFunction

A CFunction can accept up to ten arguments and returns a 64-bit integer. All arguments passed to the CFunction 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 cannot be used.

There are a few points to note:

- The CFunction returns a value which is a 64-bit integer (a long long int in C).
- 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 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 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.
- Arrays are passed by using the array name with empty brackets (ie, with no dimensions). For example: `x = CFun(a%, b%(), c%)`. In the CFunction the argument will be a pointer to the first element of the array.

A CFunction module has some important restrictions:

- 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 CFunction 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").
- It is hard to debug CFunction modules and most programming errors will cause a MIPS processor exception which in turn will instigate an immediate reboot of the Micromite.

Compiling a CFunction

It is strongly recommend that you read through the CFunction tutorial written by Peter Carnegie and use his MMBasic CFunction Maker program to convert the output from the C compiler into an MMBasic CFunction. This program adjusts for some non-position independent code that might be generated by the compiler and formats the data correctly.

The program and tutorial can be downloaded from: http://www.g8jcf.dyndns.org/mmbasicii/

The following is intended only for people who do not wish to use Peter's program or are interested in the mechanics of how a CFunction module is created.

The module should be compiled using the Microchip MPLAB X and XC32 compiler (both available for free from http://microchip.com). It must be compiled using the options:

```
-fPIC –mno-abicalls.
```

After a successful compile the disassembled output of the program can be displayed by selecting *Window* → *Output* → *Disassembly Listing File* within MPLAB X

The listing below shows a typical disassembled output:

```
Disassembly Listing for CFunctionTest
Generated From:
T:/MMBasic_Plugin.X.production.elf
12/09/2014 9:30:23 PM

---  t:/mmbasic_plugin.x/main.c  ------------------------------------
17:                     long long int main(long long int *a, long long int *b) {
9D0020DC  27BDFFF8    ADDIU SP, SP, -8
9D0020E0  AFBE0004    SW S8, 4(SP)
9D0020E4  03A0F021    ADDU S8, SP, ZERO
9D0020E8  AFC40008    SW A0, 8(S8)
9D0020EC  AFC5000C    SW A1, 12(S8)
18:                         return *a + *b;
9D0020F0  8FC20008    LW V0, 8(S8)
9D0020F4  8C440000    LW A0, 0(V0)
9D0020F8  8C450004    LW A1, 4(V0)
9D0020FC  8FC2000C    LW V0, 12(S8)
9D002100  8C460000    LW A2, 0(V0)
9D002104  8C470004    LW A3, 4(V0)
9D002108  00861021    ADDU V0, A0, A2
9D00210C  0044402B    SLTU T0, V0, A0
9D002110  00A71821    ADDU V1, A1, A3
9D002114  01032021    ADDU A0, T0, V1
9D002118  00801821    ADDU V1, A0, ZERO
19:                     }
9D00211C  03C0E821    ADDU SP, S8, ZERO
9D002120  8FBE0004    LW S8, 4(SP)
9D002124  27BD0008    ADDIU SP, SP, 8
9D002128  03E00008    JR RA
9D00212C  00000000    NOP
```

The machine code for the CFunction is shown highlighted in yellow. Each group of eight hex digits represents one 32 bit word and each must be copied into the BASIC program.

Defining the CFunction in MMBasic

The machine code is inserted into the BASIC program as a sequence of 8-digit hex words between CFUNCTION and END CFUNCTION commands. Each word must be separated by one or more spaces or a new line and consists of eight hex digits representing the bits in a single 32-bit word.

The first word of the CFunction data must be the offset (in 32 bit words) to the entry point for the program from the start of the program. In the above case this is zero but when multiple functions are used smaller functions may be placed by the linker before the main program and this is how MMBasic can find the main entry point.

When a BASIC program is saved to flash MMBasic will search through it looking for any CFUNCTION 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 name is encountered (it is erased when a new program is saved).

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

The following shows a CFunction module built from the disassembly listing shown on the previous page (the highlighted data):

```
CFunction TestCFunct
   00000000
   27BDFFF8 AFBE0004 03A0F021 AFC40008 AFC5000C 8FC20008 8C440000
   8C450004 8FC2000C 8C460000 8C470004 00861021 0044402B 00A71821
   01032021 00801821 03C0E821 8FBE0004 27BD0008 03E00008 00000000
End CFunction
```

Note that the first word is the offset to the entry point to the CFunction and is not part of the disassembled output. It is zero in this case because we are only using one function and it is first in the module.

To call the CFunction in your BASIC program you use the CFunction name (specified after the CFunction command) in an expression. The following shows the full BASIC program calling the example CFunction (which simply adds two integers and returns the result):