

Quick Start User Guide  
Installing and Running  
For GNU 'C' Users

## ■ Installing & Running

The Quick Start documents will form a basic introduction to using the FlashModule product range. They are intended for those who prefer to experiment rather than have to read a weighty document to understand how to use a new system. From these documents you will see just how easy it is to use the FlashModule for your particular application.

For many users these may be the only documents that you need to read. For users who want the more advanced documentation this is included on the shipping CD in Adobe Acrobat format. The Quick Start documents should be read as you are performing the tasks described and it forms a tutorial in using the system, from unpacking the system and installing the software up to developing your own turnkey program running from the Flash memory when power is applied to the board.

Through out the documentation the word 'module' will be used as a generic name to refer to a software module in the FlashModule's memory map that has valid identifiers (header and checksum). They can be programs, data tables, system functions, drivers or device initialisation tables.

The system is case sensitive, a program called 'Hello' is not the same as one called 'hello'. Make sure that you are typing filenames and program names in the correct case otherwise an error message will be reported.

In this document screen output will always be shown in a *courier font like this*. Commands that you should type are shown in an **Arial font like this**.

The characters **(CR)** are used to represent the 'carriage return' or 'enter' key.

The characters **(ESC)** are used to represent the 'Esc' key.

The characters **(CTRL)** are used to represent the 'Ctrl' key.

The characters **(ALT)** are used to represent the 'Alt' key.

The characters **(F10)(CR)** are used to represent the 'F10' and the 'Enter' keys pressed and released individually in sequence

The characters **(CTRL+F10)** are used to represent the 'Ctrl' key pressed then the 'F10' key pressed and released then the 'Ctrl' key released.

The characters **(ALT)t1** are used to represent the 'Alt' key pressed and released then the 't' key pressed and released then the '1' key pressed and released.

1

## ■ Unpacking and Connecting up

The FlashModule shipping package should contain the following items. If any are missing, please contact your distributor immediately.

- Quick Start Guide 1 and 2
- FlashModule controller
- CD Containing Software Support, C Compiler & Documentation
- 5 Volt supply, mains powered
- Serial Interface Cable
- Flash Formatter (Multi copy packs only)

Carefully remove the FlashModule from its anti-static bag observing anti-static precautions. Make sure that the power supply provided is suitable for connection to your mains supply. In a development environment the FlashModule range requires a single 5 Volts d.c. supply at up to 500mA. Connect the COLOURED wire to the terminal marked '5V' and the BLACK wire to the terminal marked '0V'. Connect the serial lead provided to the 10 way connector labelled 'S1' and the other end to a serial port on your PC. Before going any further check that the above has been completed correctly.

## ■ The First Application

Before the FlashModule is shipped out a demonstration program is loaded into the Flash memory on the board and set to run when the power is applied (turnkey). The first thing that we are going to do is run this demonstration program. This program checks that the FlashModule is functioning correctly and demonstrates some of the features of the range. The first thing to do is set up a terminal program on your PC. For the demonstration this terminal could be the terminal program provided with Windows or Hyper-Terminal, which is supplied with Windows 95. The terminal program should be configured with hardware handshaking at 38400 baud, 8 data bits, 1 stop bit and no parity on your chosen COM port. Having configured your terminal check that the switch in the middle of the FlashModule is in the 'R' un position. This means that when the FlashModule powers up it will start to run the turnkey program. Turn the power supply on. The green LED, D1, next to the power connector should light up and the LED, D2, next to the processor will flash briefly. The demonstration program will now start to run. The demonstration program will do the following:

1. Flash the LED D2.
2. Run a RAM test on system RAM
3. Verify that the Flash Memory is functioning correctly
4. Print the current time and date
5. Display a number of messages on the screen

Once this is completed you can reposition the switch SW2 in the 'D' ebug position and turn the power off. Close down the terminal program. You are now ready to install the software to generate your own program to run in turnkey mode. Please follow the following steps carefully.

## ■ Installing and Setting up the Software

If you are using either Windows 2000 or Windows NT please ensure that you are logged onto your computer as an administrator before attempting to install the software.

To install the software insert the CD into the CD player on the PC. If you have the feature enabled the CD will auto run and a document will be opened in your browser. If the auto run is not enabled run the program 'AUTORUN.EXE' in the root directory on the CD. The document displayed in the browser acts as a menu to the installation process which is split into three separate parts.

First install the GNU Tools. To do this, select the underlined link 'GNU Tools'. This will bring up another page that tells you more about the tools that you are about to install. To install the software click on the underlined link 'Install GNU Tools'. This will bring up a window in the browser with two options. Choose 'Open it'. Depending on your version of browser you may receive a warning at this point. Please choose to continue the installation. This will run the familiar 'InstallShield' wizard to install the software onto your machine. From this point follow on screen prompts. We strongly recommend that you do not change the directory name for this installation. You do not need to restart your system until you have installed the complete package. Use the browser 'back' key to return to the main menu.

Next you can install the documentation from the install index in the same way by following the link 'Install Documentation'. If you do not have an Adobe Acrobat reader installed you will also need to install this package. The final part of the installation is to install the CMS GNU Support Package. To do this click on the link 'FModule Support' in the main installation menu. Repeat as for the GNU installation. To complete the setup you must restart your computer.

## ■ A Simple Example

Having completed the installation we can now make sure that everything is working with a few simple examples. In this example we are going to see how easy it is to load and run a program. There is a program in the examples sub directory of the installation directory called 'led.c'. This program has already been compiled and linked to a module called 'led'. Make sure that the mode switch on the FlashModule is in the 'D' ebug position. To run the CMS terminal program, 'Target', click the 'Start' button on your desktop and choose 'Programs' -> 'FModule' -> 'Target'. The terminal starts up using COM1. If this is not correct type **(ALT+F3)** and select the correct serial port. Hit the **(CR)** key a couple of times and you should get the C> prompt on the screen. If not, check that the PC is connected to the correct serial port on the FlashModule and that the FlashModule is powered up (green LED will be on). Make sure that the terminal program is running in the correct directory (it is displayed on the title bar – to proceed it needs to be 'C:\FModule\Projects\Examples'. If it is not, on the command line type

```
cd \FModule\Projects\Examples(CR)
```

Now at the prompt type

```
load led(CR)
```

Remember the system is case sensitive. The FlashModule will now download the program 'led' from the PC. This program can be run simply by typing its name ie:

```
led(CR)
```

Notice that the red LED D2 is now flashing. Hit the **(SPACE)** key to stop it and return to the command line prompt. To run this program from reset or power on we must first place the module in battery backed memory and then turnkey it by typing:

```
unload TurnKeyD(CR)
```

```
lock led (CR)
```

```
turnkey led(CR)
```

This will create a data module which is in battery backed RAM. If the switch SW2 is in the 'R' un position when the FlashModule starts up it will run the turnkey program. Go ahead and try it. Turn the power off, then turn it back on. The LED program is now running as your application would run. You can now see how easy it is to get an application up and running. To return to the system simply set the switch back to the 'D' ebug position and reset the controller again. When you have done this, remove the battery backup from the turnkey module by typing:

```
unlock TurnKeyD(CR)
```

Next time the FlashModule is initialized the TurnKeyD module will not be found so the FlashModule will run the default terminal program.

You can now unlock the 'led' program in the same way.

```
unlock led(CR)
```

## ■ Erasing the Flash

To erase the user sectors of the Flash memory a utility program is included called 'erasef'. This program will remove all modules and data that are contained in the selected flash sectors and leave them blank ready to be reprogrammed. It should be used with care. The system sectors (addresses in the range \$000000 to \$2FFFFF) can not be erased with this command. To erase all user sectors of the Flash memory type:

```
erasef(CR)
```

You will then be asked if you want to 'Erase all User sectors' or 'Choose which sectors' you erase. This time choose option '1'. If you were to select the second option you will be asked to enter the address of the first sector. This must be greater than \$30000 for the erase function to occur. You will then be asked how many sectors you want to erase. If you do not have a formatter plugged in, the upper limit is 5, with the formatter fitted up to 14 sectors can be erased. Close the terminal program down by typing **(ALT+F10)** while the terminal program is selected.

## ■ Useful Debug Commands

The FlashModule has an integrated debugger/monitor. To enter debug mode, on the command line, type **d(SPACE)(CR)**. Once in the debugger you can use the following commands to check the status of the processor and memory.

```
dump d [address](CR)
```

display a page of memory starting at the given address

```
change c [address](CR)
```

display and modify data at a memory location (use with care)

```
registers r(CR)
```

display the current value of the processor registers

```
trace t(CR)
```

execute a single instruction and then wait for further input

For full details on all the debugger commands and their abbreviations please refer to your C User Manual. You may often see debugger commands written as 'c c F00000' for example. The above command will enable you to change the byte at address \$F00000 from the shell command line. The debugger will be called and the string following the 'd' is passed directly onto the debug monitor.

### IMPORTANT

Cambridge Microprocessor Systems Ltd. offer technical support by fax, email and Internet. Many common questions are answered on our web site at [www.cms.uk.com/support.html](http://www.cms.uk.com/support.html). This site has recently been updated to include solutions to common problems found during the installation and some extra example programs for the GNU compiler. If these do not resolve your questions the support fax number is +44 (0)1 371 876 077 and the email address is [support@cms.uk.com](mailto:support@cms.uk.com). Please freely use these support facilities if required. We offer free unlimited technical support via these services.

Quick Start User Guide  
 Compiling and Debugging  
 For GNU 'C' Users

■ **Compiling and Debugging a Program**

This section will show you how to compile a program from scratch and then load it into the FlashModule. It will also introduce some of the debugger commands that are available to help resolve programming problems.

Click the 'Start' button and choose Program -> FModule -> CEdit. This will run the CEdit program which will bring up an editor window. Select menu option 'Project' -> 'New'. Give the project a name, for example 'hello' and click on 'OK'. You will see a C program appear in the window. The editor has created a new project called 'hello' in the 'Projects' sub-directory and written a simple program for you called 'hello.c'. It also automatically generates a makefile to instruct the compiler how to build your program. All the program does is print a series of numbers on the screen.

To turn this simple C program into an application program to download to the FlashModule simply select 'Project' -> 'Build'. This will compile the program, link it, add on the Minos header and generate a symbol table to aid debugging. When the build is complete, check for any errors displayed in the bottom window. Next select 'Target' from the menu bar, followed by 'Select' to choose your target card (you only need to do this once), then click on 'Connect'. You are now running the CMS terminal program as before, which will allow you to communicate with the FlashModule. Notice that the current directory for the terminal program is now your current project directory which would be C:\FModule\Projects\hello if you have chosen all the default options.

Before turning on the power supply to the FlashModule make sure that the switch SW2 in the middle of the board is in the 'D'ebug position. Turn the power to the FlashModule on. When the board has powered up a message will be displayed followed by the prompt

```
C.M.S. Cross Compiler Support
C >
```

We now need to download the program and the symbol table that we have just created. To do this simply type:

**load hello hello.sym(CR)**

This will load the program 'hello' into the RAM on the FlashModule. When it comes to the symbol table the program will ask if you want to load it into RAM or flash. At the moment just type 'r' to load the symbol table into RAM. If you now type the command:

**mdir(CR)**

The module directory will be displayed (see below). You can see that there are a number of programs already loaded in the FlashModule's memory. It gives some useful information about all the programs in the memory map. This information is

- Start address of the module's header
- Size of the Module (not including the header)
- The name of the module
- The type of the module
- What type of memory it is located in

ADDRESS	SIZE	MODULE NAME	TYPE	MEMORY
0057a2	34	sysinit	System	Flash
0057f8	50	ticker	System	Flash
00586a	1ce	drdisk	Driver	Flash
005a5a	2a	D1	Dit	Flash
005aa6	682	dr68681	Driver	Flash
00614a	24	TERM	Dit	Flash
006190	277c	Maths	Subrtn	Flash
00892e	722	dr8530	Driver	Flash
009072	22	S1	Dit	Flash
0090b6	22	S2	Dit	Flash
0090fa	27da	shell	Absolute	Flash
00b8f6	4afc	load	Absolute	Flash
010414	22b0	procs	Absolute	Flash
0126e6	27d8	mdir	Absolute	Flash
014ee0	207c	echo	Absolute	Flash
016f7e	1f06	code	Absolute	Flash
018ea6	2cec	unload	Absolute	Flash
01bbb4	21ec	lock	Absolute	Flash
01ddc2	21f2	unlock	Absolute	Flash
01ffd6	34d4	setime	Absolute	Flash
0234cc	36d2	erasef	Absolute	Flash
026bc0	2222	turnkey	Absolute	Flash
028e04	25a4	save	Absolute	Flash
f00000	19d4	hello	Absolute	Ram
f441c0	664	hello.sym	Data	Ram

The two items in this list that we are interested in at the moment are those called 'hello' and 'hello.sym'. You can see that 'hello' is loaded at address \$F00000 in Ram and it is of type 'Absolute'. This means that the load address is included in the module header and it must always be loaded and run from this address. The module 'hello.sym' is loaded at address \$F441C0 (may be different on your controller) in the Ram and it is a 'Data' module. The symbol table name will always be in lower case, your program name will be the same as the project name. A data module can only store data, it can not be run and it is up to the application program how to interpret the contents of the data module. Next, to tell the debugger the name of the required symbol table we wish to use we need to type the following:

**d name hello(CR)**

to set a break point at the label 'main' type:

**d b main(CR)**

Next we can run the program by typing:

**hello(CR)**

The program will start running and then print the message:

```
Stopped at break
=>
```

You are now in the debugger. To see the instructions that are to be executed next type

## di(CR)

The display should look similar to the following:

```
=>di
main link a6, #ffff
main+000004 jsr __main
main+00000a clr.l fffc(a6)
main+00000e moveq #09, d1
main+000010 cmp.l fffc(a6), d1
main+000014 bge.s main+000018
main+000016 bra.s main+000030
main+000018 move.l fffc(a6), -(a7)
main+00001c pea gcc2_compiled.
main+000022 jsr printf
main+000028 addq.l #8, a7
main+00002a addq.l #1, fffc(a6)
main+00002e bra.s main+00000e
main+000030 moveq #00, d0
main+000032 bra main+000036
main+000036 unlink a6
main+000038 rts
__main link a6, #0000
__main+000004 unlink a6
__main+000006 rts
exit link a6, #fff8
exit+000004 move.l d3, -(a7)
exit+000006 move.l d2, -(a7)
```

Hit the **(ESC)** key and then type **t(CR)** to trace to the next command. To continue tracing simply hit **(SPACE)**. When you have finished tracing the program hit the **(ESC)** key again. This will stop the trace and return the debugger to the debug prompt. At the **'=>'** prompt type **g(CR)** to continue executing the program.

The message

```
The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9
C >
```

will be displayed on the screen. This is all that is required to compile and run any program. Next, close the terminal down by typing **(ALT+F10)** and then editor window down by typing **(ALT+F)X**. Finally, turn the power off to the FlashModule.

## ■ Programming into Flash

Once you have finished developing the application program you will want to include it in the Flash memory on the FlashModule so that it is permanently saved. The following procedure takes you through the steps required to place a program into the Flash memory.

We are going to place the example program 'hello.c' that we were working with previously into the Flash memory. First of all the program needs to be compiled and linked for an address in the Flash memory. To do this make sure that you have the program in the current editor window. Open up the makefile for the project and make the following changes

1. In the line LFLAGS change the text m68k-coff.x to flash.x
2. In the line sbin change the /O=F00000 to /O=30000

Save the makefile and return to the program window. To ensure that the program is rebuilt, save the file and then select 'Build' from the 'Project' menu. The program will be compiled and linked for the new address in Flash memory. When the build is completed open a 'Target' window or select your current target window. Make sure that the power to the FlashModule is on. If you hit the **(CR)** key you will get the prompt:

```
C>
```

At this prompt type the following:

## load hello(CR)

This time the program will display the size of the program and report that it is erasing the Flash and then start to copy the program 'hello' into the Flash memory. If you type:

## mmdir(CR)

you will see that the program 'hello' is located at address \$30000 in the Flash memory, compare it with the module directory that you generated earlier. You can now run the program simply by typing its name ie:

## hello(CR)

This will print the same message on the screen as we saw earlier when we ran the same program. This program is now in Flash and it can only be removed by 'unload'ing it or erasing the Flash sector which contains it.

## ■ Running from Power On

We have already seen how to turnkey a program, but in the previous example we kept the turnkey program in battery backed RAM. This time we are going to place both the program and the TurnKeyD module in the Flash memory. We have already got the program 'hello' in the Flash from the previous exercise.

Type:

## turnkey hello(CR)

This will create a data module in battery backed RAM called TurnKeyD. Move the switch SW2 to the 'R' un position and then press and hold the reset button on the FlashModule until the red LED D2 blinks, you can then release the button. This time when the board recovers from the reset, your loaded program will run. Our program has been run from reset as we saw before. Move the switch SW2 back to the 'D' ebug position and reset the FlashModule once again. This time the terminal environment is restored. To make this TurnKeyD module more permanent it can be loaded into the Flash memory. To do this, at the command prompt type:

## save TurnKeyD(CR)

This will save the data module to your hard disc. Next the battery-backed module must be removed by typing the following:

## unlock TurnKeyD(CR)

## unload TurnKeyD(CR)

The new module can then be loaded into the Flash memory by typing:

## load TurnKeyD(CR)

and when asked if you want to load the module into Ram or Flash answer by typing **F**. This will load the data module into the Flash memory. It is now permanent and can only be removed by 'unload'ing it or erasing the flash sector containing it. You can now move the mode switch SW2 to the 'R' un position. Reset the FlashModule by either turning the power supply off and then on again or by using the reset button. The FlashModule will power up and display our messages as a complete standalone system. You should now be ready to begin developing your own program. For further information on this product please refer to the main product manuals.

### IMPORTANT

Cambridge Microprocessor Systems Ltd. offer technical support by fax, email and Internet. Many common questions are answered on our web site at [www.cms.uk.com/support.html](http://www.cms.uk.com/support.html). This site has recently been updated to include solutions to common problems found during the installation and some extra example programs for the GNU compiler. If these do not resolve your questions the support fax number is +44 (0)1 371 876 077 and the email address is [support@cms.uk.com](mailto:support@cms.uk.com). Please freely use these support facilities if required. We offer free unlimited technical support via these services.