

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

■ Introduction

The Quick Start documents will form a basic introduction to using the QuickFIRE controller. It is intended for those who prefer to experiment rather than have to read a lengthy document to understand how to use a new system. From this document you will see how easy it is to use the QuickFIRE for your particular application.

For many users this may be the only document 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 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 QuickFire processor's memory map that has valid identifiers (header and checksum). They can be programs, data tables, system functions, drivers or device initialization 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.

Throughout this manual screen output will always be shown in a **courier font like this**. Commands that you should type in 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' and 'F10' keys pressed and released together.

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.

■ Unpacking and Connecting up

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

- Quick Start Guide 1, 2 and 3
- QuickFire controller
- CD Containing Software Support, C Compiler & Documentation
- 5 Volt supply, mains powered
- Serial Interface Cable

Carefully remove the QuickFire from its anti-static bag observing anti-static precautions. Make sure that the power supply provided is suitable for connection to your mains supply. If it is not the QuickFire is shipped configured for use with a

single 5 Volts d.c. supply at about 50mA to function. Connect the COLOURED wire to the terminal marked '+' and the BLACK wire to the terminal marked '0V'. Connect the serial lead provided to the 10 way connector labeled 'PL6' 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 QuickFire is shipped 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 QuickFire 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 HyperTerminal, which is supplied with Windows 95 and above. 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 option switches SW2 are set as follows -

A	On
B	On
C	Off
D	Off

on the QuickFire. This means that when the QuickFire powers up it will start to run the turnkey program. Turn the power supply on. The LED, D1, next to the power connector should light up and the LED, D4, near to the processor will flash briefly. The demonstration program will now start to run. The demonstration program will do the following:

- Flash the LED D4.
- Run a RAM test on system RAM
- Verify that the flash memory is functioning correctly
- Print the current time and date
- Display a number of messages on the screen

Once this is completed you can reposition the mode switch to SW2 bank B to the 'Off' position and reset the controller using the reset button (SW1).

When the command prompt returns we are going to erase the Flash memory so that the demonstration program has been removed and the memory is ready for us to continue developing. To do this on the command line type

```
erasesf(CR)
```

The program will ask if you want to erase all user sectors, or choose a sector to erase. We want to erase all user sectors so select this option. When the process is complete 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

Before Installing the software, make sure that you have Administrator permissions on your PC. 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 autorun is not enabled run the program 'AUTORUN.EXE' in the root directory on the CD. The document displayed in your default internet browser acts as a menu to the installation process which is split into three separate parts.

■ Installing GNU Tools

First of all 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 'GNU Tools' once more. This will bring up a window in the explorer 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. After the 'splash screen' you will be asked to accept the GNU General Public license before you can continue. The program will then prompt you with the Destination folder where the files will be placed. You can change the drive letter but we strongly recommend that you accept the directory structure. Click on 'Next' to continue. The software will now start to install onto your hard drive in the chosen folders. This may take some time depending on the speed of your CD player and your computer. When the installation is complete you will return to the GNU Tools installation page. Click 'Back' on your browser to return to the main installation page.

■ Installing the QuickFIRE Support Package

To install the QuickFIRE Support Package, make sure that you are in the main index page and then select the 'quickFire' link. The next screen will give you some further information on the package that you are about to install. Click on 'Install quickFire Support' to start running the installation program. Choose to 'Open' the program and then, if prompted, to continue. When the InstallShield has loaded, you will then be asked to accept the Minos License agreement before you can install the software. You will then be asked where the executable programs should be installed (this is not your working directory), this will normally be within the 'Program Files' folder. Click on 'Next' to continue. You are now asked where you would like to install the QuickFIRE support, this is your default working directory. Click on 'Next' to continue. The software will now be installed. When the installation is completed, you will be prompted to restart your computer. If you want to restart now, click on 'Finish'. If you want to restart later, select 'No' and then click on 'Finish'.

All the documentation for this product is in Adobe Acrobat format. You will require Adobe Acrobat Reader version 4 or later to read this. If you have not already got a copy of Adobe Acrobat Reader you can either install the version from our shipping CD (from the main menu select 'Documentation' and then 'Install Acrobat Reader) or obtain a free copy from Adobe's web site (<http://www.adobe.com/products/acrobat/readstep2.html>). All of our documentation will be installed onto your computer during the software installation.

■ SimpleExample

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 SW2 bank B on the QuickFire is in the 'Off' position.

To run the CMS terminal program, Target go to the click on the 'Start' button on your desktop. This will bring up a menu. Move the mouse up to 'Programs' and then go across. A new menu will appear with all your programs on. Find in this menu the folder 'quickFire'. Move across again and select the program '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 QuickFire and that the QuickFire is powered up (LED D1 will be on). Make sure that the terminal program is running in the correct directory (it is displayed on the title bar - should be 'C:\quickFire\Projects\Examples'. If it is not, on the command line type

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

Now at the prompt type

```
load led(CR)
```

Remember the system is case sensitive. The QuickFire will now download the program 'led' from the PC. To run this program type in its name ie:

```
led(CR)
```

Notice that the red LED D4 is now flashing. Hit the (ESC) 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:

```
lock led (CR)  
turnkey led(CR)
```

This will create a data module which is in battery backed RAM. If the mode switch SW2 bank B, is in the 'On' position when the QuickFire starts up it will run the turnkey program. Go ahead and try it. Turn the power off, and 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 mode switch back to the 'Off' position and reset the controller again. When you have done this, remove the battery backup from the turnkey module by typing:

```
unlock TurnKeyD(CR)  
unlock led(CR)
```

Next time the QuickFire is initialized the TurnKeyD module will not be found so the QuickFire will run the default terminal program. You can now unlock the

'led' program in the same way.

To continue with this demonstration of the system, please open the second Quick Start guide titled 'Compiling and Debugging'.

■ Compiling and Debugging a Program

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

Click on the 'Start' button and choose the 'Program' -> 'quickFire' -> 'CEdit'. This will run the CEdit program which will bring up an editor window. Next select the target card that you are going to develop with by selecting from the Target menu 'Target' -> 'Select' and choosing your controller from the scroll list. This sets up the computer environment for your development. 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'. 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 QuickFire simply select the 'Project' menu and option '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. Next select 'Target from the menu bar. You are now running the CMS terminal program as before, which will allow you to communicate with the QuickFire. Notice that the current directory for the terminal program is now your current project directory, which would be c:\quickFire\Projects\hello if you have chosen all the default options.

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. 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. 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 QF 'C' Users

2

Before turning on the power supply to the QuickFire make sure that the mode switch SW2 bank B is in the 'Off' position. Turn the power to the QuickFire 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 QuickFire. 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. This should look similar to the following:

Address	Size	Module Name	Type	Memory
0052ac	34	sysinit	System	Flash
005302	62	ticker	System	Flash
005386	256	rtc8583	System	Flash
0055fe	1ce	drdisk	Driver	Flash
0057ee	2a	D1	Dit	Flash
00583a	6aa	dr68328	Driver	Flash
005f06	24	TERM	Dit	Flash
005f4c	22	S0	Dit	Flash
005f90	277c	Maths	Subrtn	Flash
00872e	2744	shell	Absolute	Flash
00ae94	49b8	load	Absolute	Flash
00f86e	2262	procs	Absolute	Flash
011af2	2784	mdir	Absolute	Flash
014298	2034	echo	Absolute	Flash
0162ee	1ecc	code	Absolute	Flash
0181dc	3254	unload	Absolute	Flash
01b452	21b2	lock	Absolute	Flash
01d626	21b8	unlock	Absolute	Flash
01f800	34ce	setime	Absolute	Flash
022cf0	333a	erasef	Absolute	Flash
02604c	21f2	turnkey	Absolute	Flash
028260	2530	save	Absolute	Flash
02a7b2	1084	cd	Absolute	Flash
f00000	1dcc	hello	Absolute	Ram
f447c0	506	hello.sym	Data	Ram

C >

You can see that there are a number of programs already loaded in the QuickFire'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

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 \$F447C0 (may be different on your controller) in the Ram and it is a 'Data' module. A data module can only store data, it cannot 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)

the letter 'd' switches the QuickFire into debugger mode. All commands that are typed after this are passed to the debugger. The function 'name' will locate the symbol table initialize it. To set a break point on the label 'main' in our program we use the command 'b' as in the example below. This will cause the program to stop executing when it reaches the label 'main'

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 to trace to the next command. To continue racing simply hit the (spacebar). 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 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 QuickFire.

■ Programming into Flash

Once you have finished developing the application program you will want to include it in the flash memory on the QuickFire 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 correct project selected, go to Project -> Open and select the file 'hello.gpf' from the 'hello' folder. Open up the makefile for the project in the editor window by clicking on File -> Open and selecting the file 'makefile' and make the following changes -

```

Find the entry
'ifndef ROM
ROM=0xf00000
endif'
and change it to

```

```

'ifndef ROM
ROM=0x30000
endif' .

```

Save the makefile. Next open up the source file 'hello.c' by clicking on File -> Open and choosing the file 'hello.c'. 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 (you can only have one active 'Target' window at a time).

Make sure that the power to the QuickFire 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:

mdir(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 messages 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 'unloading' 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

turnkeyhello(CR)

This will create a data module in that is battery backed in the RAM called TurnKeyD.

Move the mode switch SW2, bank B to the 'On' position and then press the reset button on the QuickFire. This time when it powers up your loaded program will run. Our program has been run from reset as we saw before. Move the switch SW2, bank B, back to the 'Off' position and reset the QuickFire once again. This time the terminal environment is restored. To make this TurnKeyD module more

permanent it can also 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 'unloading' it or erasing the flash sector containing it. You can now move the mode switch SW2 bank B to the 'On' position. Reset the QuickFire by either turning the power supply off and then on again or by using the reset button. The QuickFire 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 please refer to the main QuickFire Documentation. The final part of these QuickStart Guides shows you how to use the source level debug.

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. 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. Please freely use these support facilities if required. We offer free unlimited technical support via these services.

Quick Start User Guide
Source Level Debug
For QF 'C' Users

3

■ Introduction.

Source Level Debug does not use the built in debug tools in the Minos operating system but uses some software that can be downloaded to the controller and some software running on the host PC.

A complete set of the documentation for GDB will be installed with the GNU tools package, however this documentation focuses on using GDB on desktop systems with large operating systems like Linux but much of the information is still relevant.

The CMS Modules use GDB to provide remote debugging using the 'GDB remote serial protocol' and a 'stub', which runs on the Module. Look in the 'Specifying a debugging Target' section of the GDB documentation for more details of remote debugging. A version of the stub program that runs as a Minos program is provided with this package, during the debug session the program under test appears to become part of the stub program.

■ Using GDB

When the CEdit development environment creates a new project the resulting makefile contains all the command line options required to generate debugging information for GDB.

A setup file called 'gdb.ini' is also created GDB reads and processes this when it starts up, other commands can be added here if required. Connect to your Module using the 'Connect' option from the 'Target' menu, and download your program in the normal way by typing –

load <program name>(CR)

Next, load in the GDB stub program 'gdbstub' by typing the following –

load \quickFIRE\lib\gdbstub(CR)

To start the debug session you need to run 'gdbstub' giving it the name of the program you will be debugging as an argument.

gdbstub <program name>(CR)

You will start to see messages that look something like this \$S05#b8, these are status messages from gdbstub. Now shut down the terminal program by pressing ALT F10 or clicking on the 'X' in the right hand corner. To start GDB from the 'CEdit' program, click on the 'Tools' menu and select the option 'GDB'. This will start GDB on your development system and read the gdb.ini file. You can add more commands to gdb.ini if you need to do any extra setup.

To connect GDB to the gdbstub program running on your module type

target remove COMx(CR)

where x is the COM port that you are using on your computer.

You will see a message like this

```
Remote debugging using comx  
0xf00041 in module_start()  
(gdb)
```

The address and COM port number will depend on which Module and which port you are using on your development PC.

At this point your program is stopped in the C start up routine, the best way to get to your C code is to set a breakpoint at the start of the main routine and start the program running again to do this type

**break main(CR)
continue(CR)**

The code will run then stop at the breakpoint and you will see a message something like this

```
Continuing.  
Breakpoint 1, main (argc=1, argv=0x193598)  
at test.c line:18  
18 for (n=0;n<10;n++)
```

The details of the message will depend on exactly what's in your source file. You can single step through the C code by typing

next (CR)

you can look at the current value of a variable, try typing

print n(CR)

Now take a look at the GDB documentation for more details of things you can do.

When you have finished debugging type 'quit' at the (gdb) prompt to close the window and return to the editor.

■ Things to watch out for

As the program program being debugged actually becomes part of the same process as the gdbstub, gdbstub will exit if the program being debugged exits (returns from main() or calls exit()). If this happens the GDB running on the host system will not notice this, to get control back press CTRL C twice. You can then start the debugging session over again.

■ Useful GDB Commands

This section gives details on some useful debugger commands. Full details can be found in the main GDB manual which is installed as part of the GNU Tools.

■ Using different Serial Ports

If you are using a controller with more than one serial port, you can run the source

level debug on a second serial port, and keep the terminal program running on the default terminal serial port. The example below uses the source level debugger on serial port S2 on a CMS controller to debug the program 'demo'.

gdbstub -p:S2 demo(CR)

Note that if the serial port S2 is not running at the same baud rate as the TERM serial port you may need to change the baud rate setting in the initialisation file 'gdb.ini'.

■ **Set a Breakpoint**

To set a break point at a particular label type the following

break <label>(CR)

■ **Run up to breakpoint**

To run the program up to the next break point type

continue(CR)

■ **Run to next line**

To run the program until it gets to the next line of source code type

step(CR)

■ **Display contents of a variable**

There are three commands to display the contents of a variable. The first two are

print <variable>(CR)

and

inspect <variable>(CR)

Both of these commands add the variable to the history. To display the contents of the variable without adding it to the history use the command

output <variable>(CR)

■ **Register dump**

To view the contents of the processor registers at any time use the command

info registers(CR)

■ **Memory Examine**

To view the contents of some memory use the command

/

for example the command

x/3xb 0xf00000(CR)

will display three hexadecimal bytes starting at address \$f00000.

■ **Change contents of a variable**

You can change the contents of any variable using the command

set <variable>=<data>(CR)

■ **Program Information**

You can display general information about the program being debugged by typing

info(CR)

This will display all the available commands. Using

info <command>(CR)

Will give information about the particular command.

■ **General Information about the Debugger**

Like the 'info' command the 'show' command can be used to get information about the debugger.

■ **Available Commands**

One of the most useful debugger commands is 'help' this will give details of all debugger commands. If it is used on its own it will list all available commands. If you follow the 'help' with a command it will give additional help on that command.

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. 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. Please freely use these support facilities if required. We offer free unlimited technical support via these services.