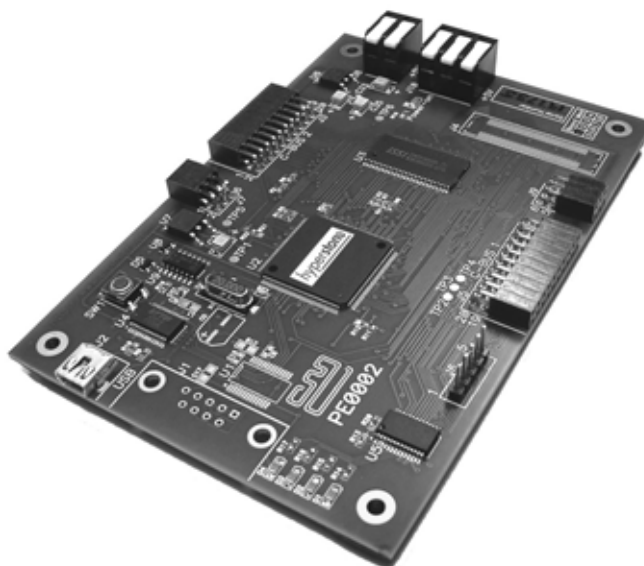


Features

- Global interface for new generation IC evaluation kits
- Target IC C-BUS read and write operations
- 32-Bit RISC/DSP Microprocessor based operation
- Mating interface for wide range of target evkit boards
- PC GUI and hardware provided
- Function Image™ handling for *FirmASIC*®-based projects
- High-speed real-time script execution
- PC control/communications via USB
- Software configurable GPIO lines and LED bank



1 Brief Description

The PE0002 EvKit interface card is a global interface system for use with evaluation kits for CML's new generation ICs, including *FirmASIC*®-based products. This greatly simplifies the approach to the evaluation and design-in process.

Based around the operation of Hyperstone's E2 32-bit RISC/DSP Microprocessor and using a PC GUI, the information generated is formatted, timed and delivered to the target IC via its Evkit's C-BUS serial interface. The supplied control software can be used to perform C-BUS read and write operations or to run scripting functions.

In the case of *FirmASIC*® IC evaluations, the Function Image™ can be loaded onto the CMX{target} device or programmed into serial memory on the {target} card.

Communications with the evaluating PC GUI are via a USB port.

CONTENTS		
<u>Section</u>		<u>Page</u>
1	Brief Description	1
2.	Preliminary Information.....	4
2.1	Laboratory Equipment.....	4
2.2	Handling Precautions	4
3.	Quick Start	5
3.1	Setting-Up	5
3.2	Adjustments	5
3.3	Operation	5
4.	Signal Lists	6
5.	Circuit Schematics and Board Layouts	9
6.	Detailed Description	10
6.1	Hardware Description.....	10
6.2	Adjustments and Controls	12
6.3	Embedded Software Description	12
6.4	Software Description	12
6.5	Evaluation Tests	22
6.6	Troubleshooting.....	22
7.	Performance Specification.....	23
7.1	Electrical Performance	23

It is always recommended that you check for the latest product datasheet version from the Datasheets page of the CML website: [www.cmlmicro.com].

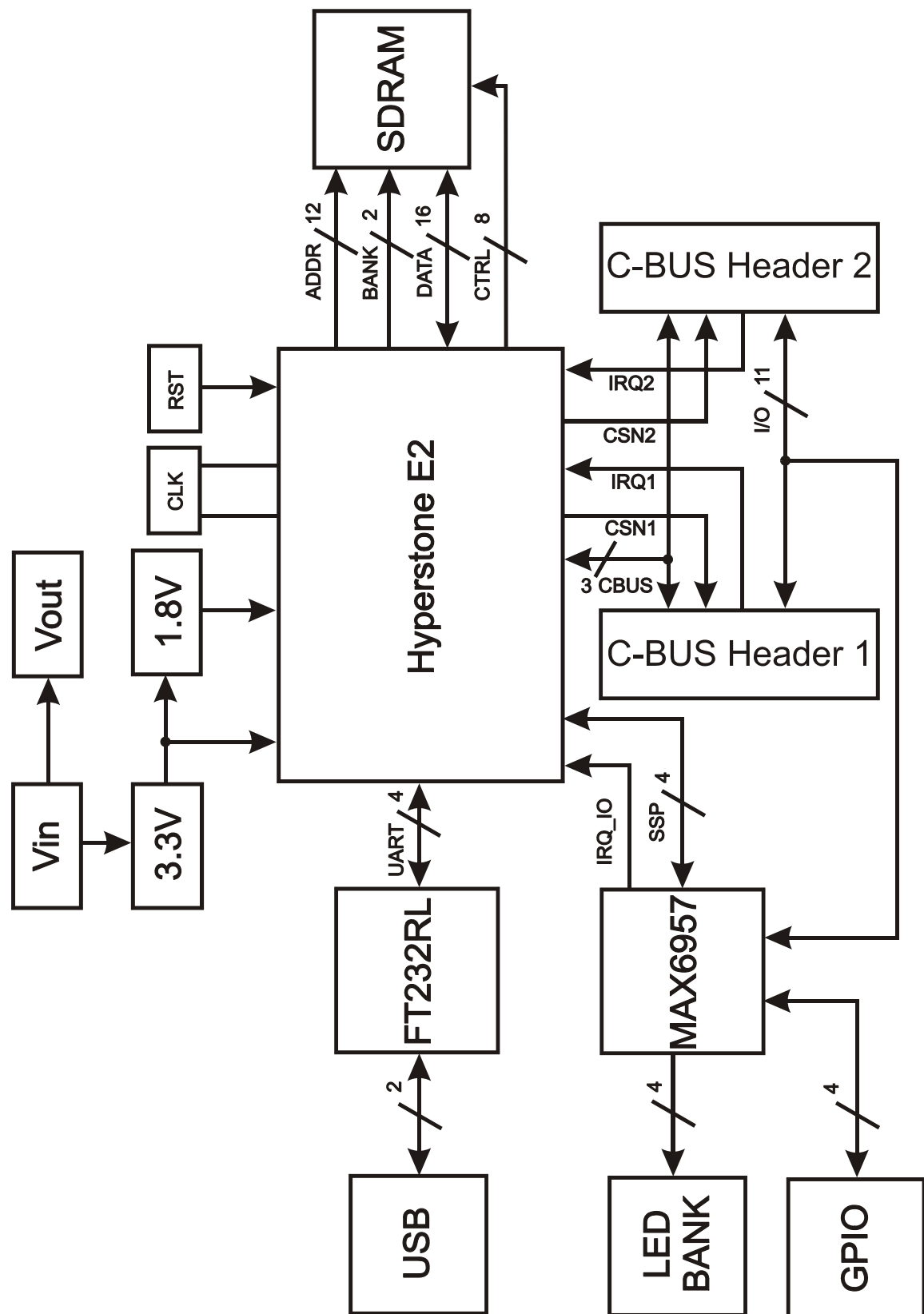


Figure 1 Block Diagram

2. Preliminary Information

2.1 Laboratory Equipment

The following laboratory equipment is needed to use this evaluation kit:

2.1.1 Power Supply

A 5 Volt dc un-regulated power supply.

2.1.2 PC

With the following requirements:

- One of the following Windows operating systems installed: 2000sp4 or XP.
- USB port.
- Minimum screen resolution 800 x 600. Recommended resolution 1024 x 768.

2.1.3 USB type A male to mini B male cable

2.2 Precautions

Like most evaluation kits, this product is designed for use in a laboratory environment. The following practices will help ensure its proper operation.

2.2.1 Static Protection

This product uses low power CMOS circuits which can be damaged by electrostatic discharge. Partially damaged circuits can function erroneously, leading to misleading results. Observe ESD precautions at all times when handling this product.

2.2.2 Contents - Unpacking

Please ensure that you have received all of the items detailed on the separate information sheet (EK0002) and notify CML within 7 working days if the delivery is incomplete.

3. Quick Start

This section provides instructions for users who wish to experiment immediately with the evaluation kit. A fuller description of the kit and its use appears later in this document.

3.1 Setting-Up

- Copy the file 'ES0002xx.zip', which is downloaded from the CML website following registration, to the hard drive of your host PC.
- Extract the files to the hard drive of your host PC.
- Connect the PE0002 Interface Card to the Target Card, via the right angle connector, J5 or J3.
- Connect a dc supply to the PE0002 Interface Card and set to the voltage level to 5V.
- Connect a dc supply to the Target Card and set to the voltage level specified in the relevant Target Card User Manual.
- Attach a USB cable between connector J2 of the PE0002 Interface Card and the USB port of the PC.
- Turn on the power supply. The power on indicator D6 will light.
- Install the USB driver when requested. The driver is in the same folder where the 'ES0002xx.zip' files were extracted in '..\Driver'. Follow instructions on the screen to install the USB driver. Click the 'Continue Anyway' button when the Message Box below is displayed.



Figure 2 USB Driver Installation Message Box

3.2 Adjustments

None.

3.3 Operation

- Run the application ES0002xx.exe.
- Press the reset switch (SW1) on the PE0002 board when requested.
- Press the 'OK' button.

4. Signal Lists

CONNECTOR PINOUT				
Connector Ref.	Connector Pin No.	Signal Name	Signal Type	Description
J2	1	VBUS		USB Mini B type connector - USB power supply
	2	D-		USB Mini B type connector - USB data signal minus
	3	D+		USB Mini B type connector - USB data signal plus
	4	n/c		
	5	GNDD		USB Mini B type connector - Digital ground
	6	SHELL		USB Mini B type connector - USB connector shell
	7	n/c		
J3	1	IO0	BI	Spare I/O
	2	CSN2	O/P	C-BUS chip select
	3	IO1	BI	Spare I/O
	4	CDATA	O/P	C-BUS command data
	5	IO2	BI	Spare I/O
	6	SCLKCBUS	O/P	C-BUS serial clock
	7	IO3	BI	Spare I/O
	8	RDATA	I/P	C-BUS reply data
	9	IO4	BI	Spare I/O
	10	IRQN2	I/P	C-BUS interrupt request
	11,12	GNDD	Power	Digital ground
	13	BOOTEN1	O/P	Hardware boot control
	14	BOOTEN2	O/P	Hardware boot control
	15	RS232/C-BUS	O/P	Hardware boot control
	16	IO5	BI	Spare I/O
	17	IO6	BI	Spare I/O
	18	IO7	BI	Spare I/O
	19, 20	n/c		

CONNECTOR PINOUT				
Connector Ref.	Connector Pin No.	Signal Name	Signal Type	Description
J5	1	IO0	BI	Spare I/O
	2	CSN1	O/P	C-BUS chip select
	3	IO1	BI	Spare I/O
	4	CDATA	O/P	C-BUS command data
	5	IO2	BI	Spare I/O
	6	SCLKCBUS	O/P	C-BUS serial clock
	7	IO3	BI	Spare I/O
	8	RDATA	I/P	C-BUS reply data
	9	IO4	BI	Spare I/O
	10	IRQN1	I/P	C-BUS interrupt request
	11,12	GNDD	Power	Digital ground
	13	BOOTEN1	O/P	Hardware boot control
	14	BOOTEN2	O/P	Hardware boot control
	15	RS232/C-BUS	O/P	Hardware boot control
	16	IO5	BI	Spare I/O
	17	IO6	BI	Spare I/O
	18	IO7	BI	Spare I/O
	19, 20	n/c		
J6	1	GPIO0	BI	Spare I/O
	2	GPIO1	BI	Spare I/O
	3	GPIO2	BI	Spare I/O
	4	GPIO3	BI	Spare I/O
	5	GNDD	Power	Digital ground
J7	1, 2	GNDD	Power	Digital ground
	3, 4, 5, 6	VCCIN	Power	+5.0V power supply
J8	1	GNDD	Power	Digital ground
	2	VCCIN	Power	+5.0V power supply

CONNECTOR PINOUT				
Connector Ref.	Connector Pin No.	Signal Name	Signal Type	Description
J9	1, 2	GNDD	Power	Digital ground
	3, 4, 5, 6	VCCIN	Power	+5.0V power supply
J10	1	GNDD	Power	Digital ground
	2, 3	VCCIN	Power	+5.0V power supply

TEST POINTS		
Test Point Ref.	Default Measurement	Description
TP1	VCC1V8	+1.8V power supply to the core of the E2 Microprocessor
TP2		Not used
TP3		Not used
TP4		Not used
TP5	CLKOUT	Clock signal output of the E2 Microprocessor
TP6	VCC3V3	+3.3V power supply

SWITCHES		
Switch Ref.	Default Position	Description
SW1	O/C	Push to reset the E2 Microprocessor

Notes: I/P = Input
 O/P = Output
 BI = Bidirectional
 O/C = Open circuit

5. Circuit Schematics and Board Layouts

For clarity, circuit schematics are available as a separate high resolution pdf file. This can be found on the CML website.

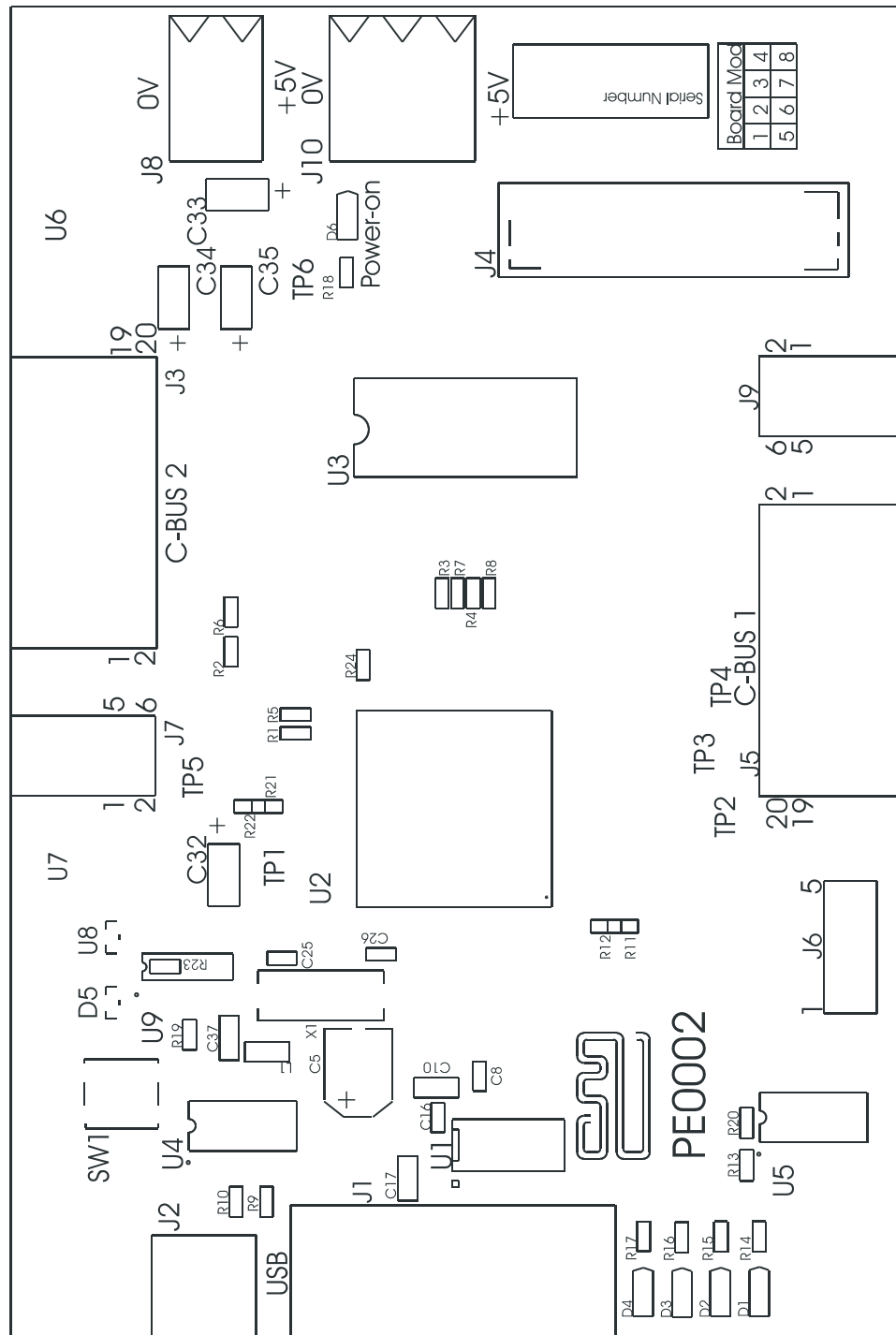


Figure 3 PE0002 Interface Card Layout

6. Detailed Description

6.1 Hardware Description

The board is fitted with two voltage regulators. U6 and U7 provide the +1.8V and +3.3V digital supply rails respectively. The input to these two regulators is provided by an external un-regulated power supply at, nominally, 5V dc, which is connected to the board via connector J8 (2 position terminal block). The external un-regulated power supply is available in connector J10 (3 position terminal block) and connectors J7 and J9 (right angle female headers)

The +1.8V and +3.3V supply voltage levels can be monitored on test points TP1 and TP6 respectively.

LED illumination confirms the presence of the on-board regulated +3.3V dc digital voltage supply.

The PE0002 Interface Card is based around the Hyperstone E2 32-Bit RISC\DSP Microprocessor. There are 32kBytes of RAM on-chip and 64Mbits of external SDRAM.

The board uses the fitted 20MHz crystal as the clock source for the E2 Microprocessor.

Two C-BUS outputs, J3 and J5, are provided for control of up to two CMX{target} devices on the selected {target} cards. The C-BUS signals, 3 boot control signals and 8 I/O signals are brought out on these connectors and can be programmed in the GUI.

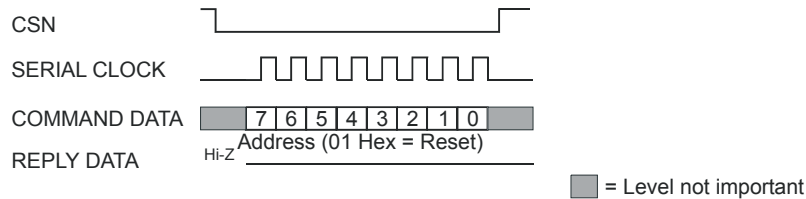
There are 4 software configurable general purpose I/O signals connected to the output J6 and 4 software configurable LEDs.

The link to a host PC is via a full speed USB 2.0 port.

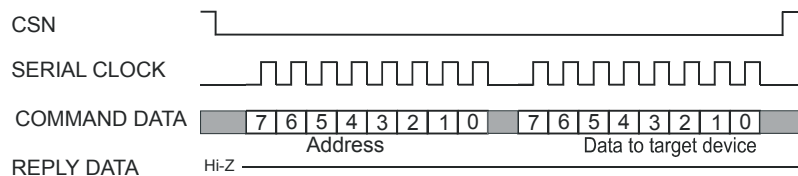
6.1.1 C-BUS Interface

The C-BUS interface provides for the transfer of data and control or status information between the target device's internal registers and the Microprocessor over the C-BUS serial bus. Each transaction consists of a single Register Address byte sent from the Microprocessor, which may be followed by one or more data byte(s) sent from the Microprocessor to be written into one of the target device's registers, or one or more byte(s) of data read out from one of the target device's registers, as illustrated in Figure 4.

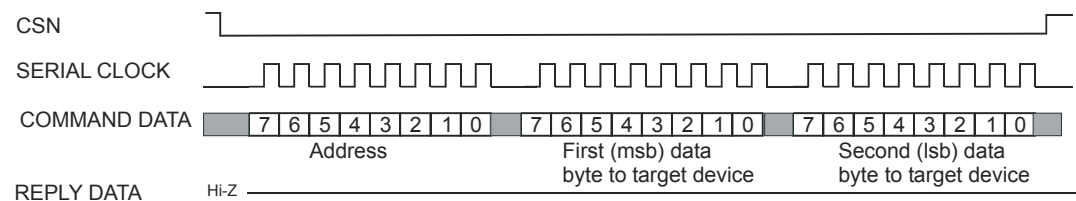
a) Single byte from μ C (General Reset command)



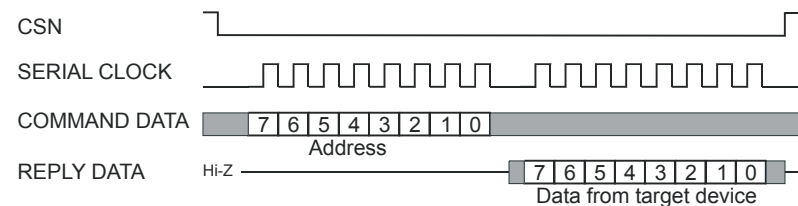
b) One Address and one Data byte from μ C to target device



c) One Address and 2 Data bytes from μ C to target device



d) One Address byte from μ C and one Reply byte from target device



e) One Address byte from μ C and 2 Reply bytes from target device

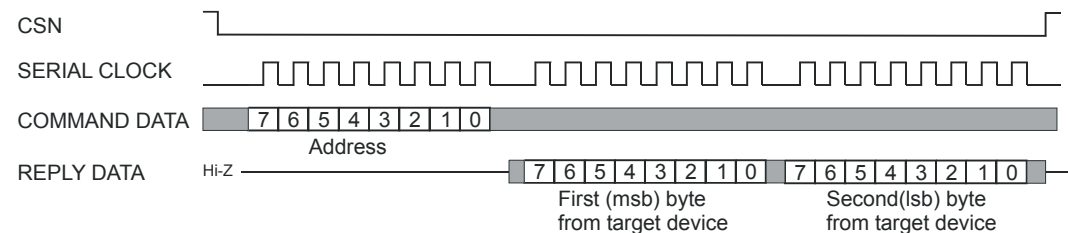


Figure 4 C-BUS Transactions

All writes and reads to the C-BUS output occur via the Serial Interface Engine of the E2 Microprocessor.

Two different CMX{target} devices can be selected by taking CSN1 or CSN2 low in connectors J5 and J3 respectively.

The CMX{target} device can issue an interrupt by taking IRQN1 or IRQN2 low in connectors J5 and J3 respectively.

6.2 Adjustments and Controls

None.

6.3 Embedded Software Description

On power up the Microprocessor will monitor the USB port to download the Firmware from the host PC to the internal RAM of the Microprocessor and will jump to the start address of the Firmware when requested from the host PC.

6.4 Software Description

A single executable 'ES0002xx.EXE' running in a Windows environment supports a range of Target Cards. Visit the CML website for the latest information.

When the ES0002xx.EXE application is run, if communication cannot be established a message box will be displayed to indicate the failure. Otherwise, the message box in Figure 5 is displayed. Press the Reset Switch SW1 on the PE0002 board and click on the 'OK' button. The Firmware is automatically downloaded to the PE0002 board and a tabbed dialog box is then displayed, which is the main application dialog.

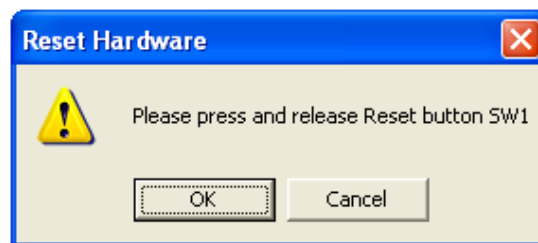


Figure 5 Start Message Box

There are four sheets within the tabbed dialog box structure and these are described in the following sections.

6.4.1 The C-BUS Control Tab

This tab provides basic C-BUS read, write and general reset functions. Each character entered into the Address and Data edit boxes is checked to ensure that it is a valid hexadecimal value. The radio buttons select an 8-bit or 16-bit read/write operation. The lengths of the entered values are validated, 2 characters (1 byte) for read or write register addresses and 2 or 4 characters (1 or 2 bytes) for the register write data. The General Reset button writes 01_H to the CMX{target} device. The radio buttons select read/write operation to the CMX{target} device 1 using CSN1 on connector J5 or to the CMX{target} device 2 using CSN2 on connector J3.

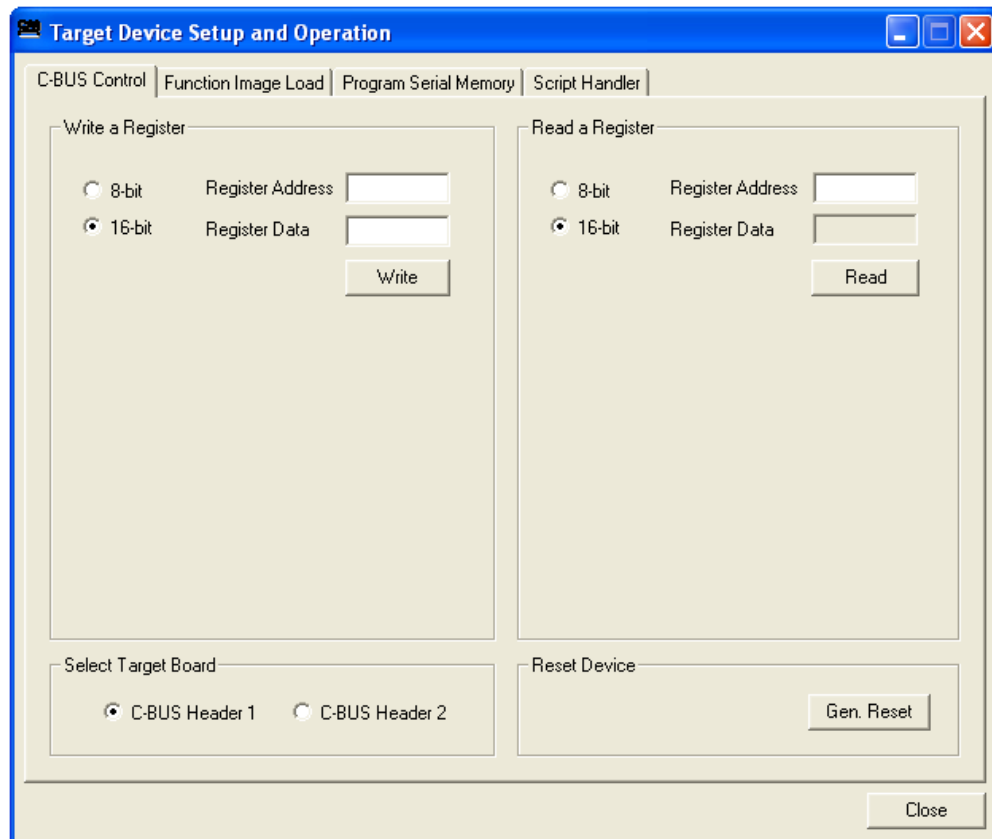


Figure 6 C-BUS Control Tab

6.4.2 The Script Handler Tab

The Script Handler tab allows the execution of scripts. These are plain text files on the PC, and are compiled by the GUI, but executed on the E2 Microprocessor on the board.

To select a script file, click on the 'Select Script' button. The Open File Dialog is displayed. Browse and select the script file. The folder that contain the script file will be the working folder of the script (i.e. all the files referenced in the script will be searched in this folder).

The results window displays the values returned by the script. These results can be saved to a text file or discarded by clicking on the 'Save Results' or 'Clear Results' buttons, respectively. When a script file is being executed the 'Run Script' button will change to be the 'Abort' button, the rest of the tab will be disabled and the other tabs cannot be selected.

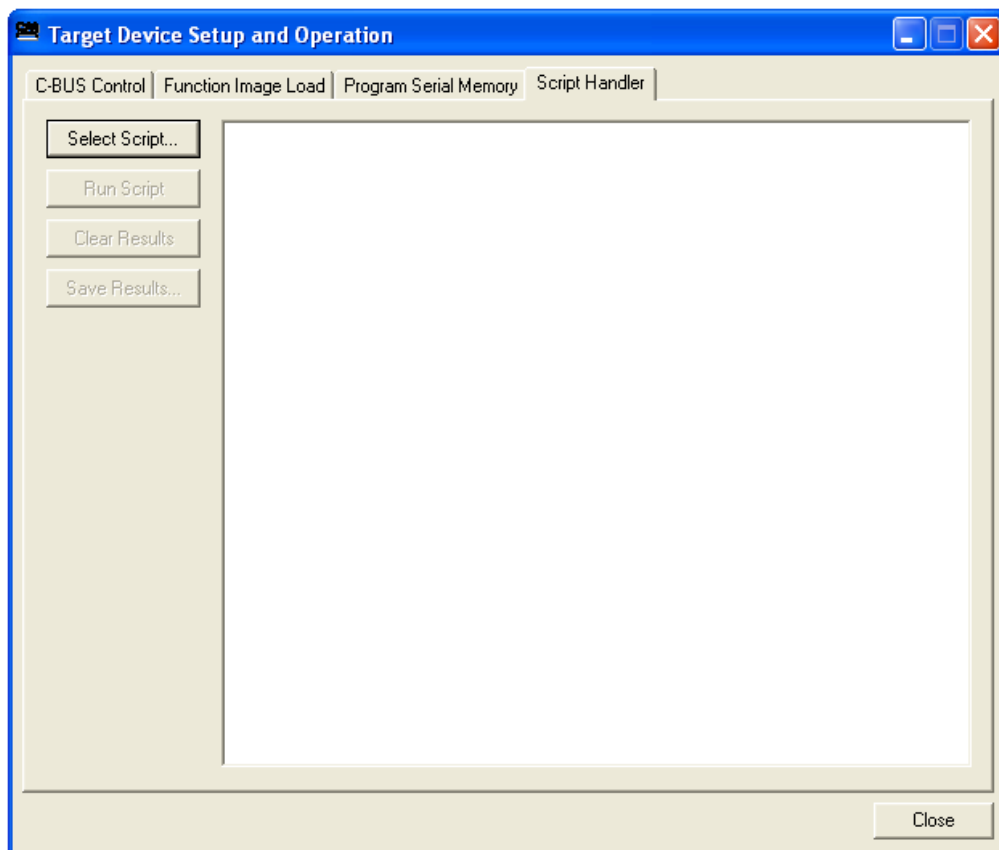


Figure 7 Script Handler Tab

6.4.2.1 Script command format

Commands take the general form of:

label command operand1, operand2,... ;comment

- Labels must start in the first column and must start with a letter or underscore.
- The command must NOT start in the first column.
- Only one command is allowed per line.
- Operands should be separated with a space or a comma. The number of operands required varies from command to command. Most commands have a fixed number of operands, but the disp, dialog and filew commands will accept one or two. Also, if and while commands have a flexible syntax.
- Blank lines, lines with just a label, just a comment or just a command (plus operands) are all allowed.

6.4.2.2 Syntax

6.4.2.2.1 Constants

#1234	Constant, decimal value 1234. Treated as 16 bit.
678	Constant, decimal value 678. Treated as 16 bit.
#\$56AB	Constant, hex value 56AB. Treated as 16 bit.
\$FE	Constant, hex value FE. Treated as 16 bit.

6.4.2.2.2 C-BUS addresses

*45 (C-BUS) Address specified in decimal, value 45
 *\$A7 (C-BUS) Address specified in hex, value A7

6.4.2.2.3 Strings

" Hello world." Strings presented in double quotes. Strings are limited to 64 characters.

6.4.2.2.3.1 Formatted strings

Some of the commands make use of 'formatted strings'. These work much like the 'printf' function in C, with the following differences:

- \n is assumed at the end of a line (use \c to continue if you don't want a new line).
- The disp, dialog & filew commands only support one parameter after the string, so you should only use one format specifier in the string.
- The standard format specifiers supported are:
 - %d (signed decimal)
 - %u (unsigned decimal)
 - %x and %X (hexadecimal)
 - %f (floating point, when used in filer command converts to a 16-bit signed value and warns if out of range).
- The usual flags (-, +, 0, #), field width and precision specifiers can be used with these.
- The following additional format specifiers are also available:
 - %b (binary) no leading zero suppression.
 - %q (Q15 fixed point)

6.4.2.2.4 Conditions

The following conditions/operators can be used with the jmpc, jsr, if and while commands:

<	Less than
>	Greater than
= or ==	Equal to
!=	Not equal to
<=	Less than or equal to
>=	Greater than or equal to
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR

6.4.2.2.5 Variables

Variables are declared as follows:

bar	word	n	Specifies a 16 bit variable bar, initialised to n.
foo	buffer	X	Specifies an array of X 16-bit words, all initialised to zero.

Note that bar & foo are labels, and must start in the first column.

Variables can be accessed in the following ways:

- bar Return the variable bar
- foo Return the first word in the array foo.
- foo[0] Return the first word in the array foo.
- foo[3] Return the fourth word in the array foo.
- foo[bar] Return the 'n+1'th word in the array foo (bar = n).
- foo[bar++] As above, and increment bar.
- foo[bar--] As above, but decrement bar.

6.4.2.2.6 C-BUS Register lengths

All C-BUS registers are assumed to be 16 bits long. To change this use the 'register' command. For streaming registers, use the 'read' and 'write' commands.

6.4.2.3 Commands

In the tables below, the following nomenclature is used:

- "str" A formatted string as described in section 6.4.2.2.3.1
- <cond> A condition code as listed in section 6.4.2.2.4
- <label> A label referencing a script line, used in jmp, jsr, if & while commands.
- #<const> A constant as described in section 6.4.2.2.1
- *<cbus> A C-BUS address as described in section 6.4.2.2.2
- <var> A variable as described in section 6.4.2.2.5
- <any> Any one of #<const>, *<cbus> or <var>

6.4.2.3.1 IO involving the PC

Command	Parameters	Description
Fopenr	"filename", "str"	The Microprocessor instructs the PC to open file "filename" for reading. The formatted string describes the line format. The PC begins piping the file content down to the micro. Script execution will pause until all of the file has been transferred or the micro's buffer is full. The buffer will continue to be automatically refilled by the PC until all of the file has been transferred.
Fopenw	"filename"	The Microprocessor instructs the PC to open file "filename" for writing.
Filer	*<cbus>	The Microprocessor reads from the buffer created by fopenr, and writes the value to the C-BUS register <cbus>.
	<var>	As above, but write to variable <var>.
Filew	"str", <any> or "str"	The Microprocessor reads a C-BUS address, variable or constant and instructs the PC to write the result to file using formatted string "str".
Disp	"str", <any> or "str"	The Microprocessor reads a C-BUS address, variable or constant and instructs the PC to display the result in the log window with formatted string "str".
Dialog	"str", <any> or "str"	The Microprocessor reads a C-BUS address, variable or constant and instructs the PC to display the result in a dialog box with formatted string "str". Script execution will pause until the dialog box is cleared by the user.
Cls	None	The Microprocessor instructs the PC to clear the log window.

6.4.2.3.2 Arithmetic and logic operations

These commands take the form: command <source>,<destination>

So the final answer always ends up in the *second* operand.

Command	Parameters	Description
Copy	<any>,<var>	<Var>=<any>
	<any>,*<cbus>	*<cbus> = <any>
Add	<any>,<var>	<var>=<var> + <any>
Sub	<any>,<var>	<var>=<var> - <any>
And	<any>,<var>	<var>=<var> & <any>
Or	<any>,<var>	<var>=<var> <any>
Xor	<any>,<var>	<var>=<var> ^ <any>
Lsl	<any>,<var>	<var> = <var> << <any>
Lsr	<any>,<var>	<var> = <var> >> <any>
Asl	<any>,<var>	<var> = <var> << <any> (Sign bit preserved)
Asr	<any>,<var>	<var> = <var> >> <any> (Sign bit preserved)

6.4.2.3.3 Streaming C-BUS

These commands allow streaming accesses between C-BUS addresses and variables. These commands take 3 parameters: a C-BUS address, a variable which is treated as the start of a buffer (i.e. data is stored in, or read from, consecutive locations starting from this one) and a 'length' parameter. The number of data bits transferred will be 'length' x the size of the C-BUS register. The data will always be stored in or read from 'length' consecutive variables, regardless of the register length. When an 8-bit register is accessed the LSByte of the variable is used.

Command	Parameters	Description
Read	*<cbus>,<var>,<any>	The Microprocessor performs a streaming C-BUS read from address <cbus> and stores the results into an array starting at <var>. <any> controls the amount of data.
Write	<var>,*<cbus>,<any>	The Microprocessor reads 'any' items from an array starting at <var> and performs a streaming C-BUS write to address <cbus>. <any> controls the amount of data.

6.4.2.3.4 Program flow

Notes:

'PC' represents the script Program Counter.

Zero equates to FALSE, any non-zero value is TRUE.

Command	Parameters	Description
Jmp	<label>	PC=<label>
Jmpc	<any><cond><any><label>	PC=<label> if <cond> is TRUE
Jsr	<label>	Push PC+1, PC=<label>
Jsrc	<any><cond><any><label>	As jsr if <cond> is TRUE
Return	None	Pop PC
If	<any><cond><any> or <any>	Jump to matching 'endif' if <cond> is FALSE
Endif	None	This is just a marker for if
While	<any><cond><any> or <any>	Jump past matching 'endwhile' if <cond> is FALSE
Endwhile	None	Jump to matching 'while'

Stop	None	Stop executing script
------	------	-----------------------

6.4.2.3.5 Miscellaneous

Command	Parameters	Description
Port	#<const>	Select Port <const> (1 or 2) to be current.
Portc	<any>	Configure current port with value of <any>. A 1 in a bit position sets the corresponding bit in the port to be an INPUT. A 0 sets it to be an OUTPUT. See below for mapping of bits to pins.
Portw	<any>	Set hardware output pins of current port to value of <any>. See below for mapping of bits to pins.
Portr	<var> or *<cbus>	Set <var> or *<cbus> to value read from hardware input pins of current port. See below for mapping of bits to pins.
Device	#<const>	Select Device <const> (Device 1 on connector J5 or device 2 on connector J3). Note: C-BUS device 1 on connector J5 is the default device.
Delay	#<const>	The Microprocessor waits for <const> milliseconds.
Register	#<const1>, #<const2>, #<const3>	Set the length in bytes <const3> of the register address <const2> in the device <const1>.

Port 1

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	GPIO3	GPIO2	GPIO1	GPIO0	DO4	DO3	DO2	DO1

b3-0 Outputs connected to LED bank. Always configured as output.
b7-4 General purpose I/O lines connected to J6. Configurable as input or output.

Port 2

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	IO7	IO6	IO5	RS232/CBUS	BOOTEN2	BOOTEN1	IRQN 2	IRQN 1	IO4	IO3	IO2	IO1	IO0

b4-0 I/O lines connected to J3 and J5. Configurable as input or output. Check user manual of the {target} card for assignment if any.
b6-5 Interrupt input lines connected to J3 and J5. Always configured as inputs.
b8-b7 Boot mode selection output lines connected to J3 and J5. Always configured as outputs.
b9 RS232/C-Bus mode selection output line connected to J3 and J5. Always configured as output.
b12-b10 I/O lines connected to J3 and J5. Configurable as input or output. Check user manual of the {target} card for assignment if any.

When the GUI is first started all the configurable I/O lines are set as inputs.

6.4.2.4 Errors

6.4.2.4.1 Build errors

Build errors always report the line number in the text file where the problem was encountered. In the case of missing `endif/endwhile` commands the line number for the unmatched `if/while` is given. The error messages are listed below:

- Unrecognised command
- `If/while` without matching `endif/endwhile`
- Unresolved label(s)
- Command requires `n` parameters
- Parameter type not allowed
- Duplicate label

6.4.2.4.2 Run time errors

Runtime errors always report the value of the PC where the problem was encountered.

6.4.2.4.2.1 PC out of range

The script program counter (i.e. the address of the next instruction) is not within the script. This probably means a `jmp`, `jsr` or `return` went wrong somewhere.

6.4.2.4.2.2 Invalid opcode

The Microprocessor is trying to execute a command it doesn't recognize.

6.4.2.4.2.3 Data index out of range

The variable addressed is outside the data array. Probably caused by excessive use of the `foo[bar++/--]` syntax.

6.4.2.4.2.4 Stack overflow / underflow

The stack is used to hold return addresses when `jsr` or `jsrc` commands are used. Overflow usually indicates repeatedly using `jsr(c)` without a matching `return`. Underflow results from using `return` without having first used `jsr(c)`.

6.4.2.4.2.5 Tx buffer overflow

This is the buffer used to transfer data from the Microprocessor to the GUI. Overflow indicates that data is being put into it too fast. Put data in more slowly. Commands that put data into this buffer include `disp` and `filew`. Note that the contents of the string are not sent over this link, so reducing the length of the strings won't help.

6.4.2.4.2.6 File buffer overflow / underflow

This is the buffer used for file data sent from the GUI to the Microprocessor. The amount of data in the buffer is monitored and managed by the Microprocessor in such a way that it should never overflow. It may underflow if the script uses the data faster than the GUI can supply it. Read data from the buffer more slowly and close any other open applications to avoid buffer underflow.

6.4.2.4.2.7 Invalid Jump Destination

This means that the value which a jmp, jsr return, if or while command is trying to write to the script program counter (i.e. the address of the next instruction) is not within the script

6.4.2.4.2.8 Attempt to reference non-existent string

This means that the Microprocessor has passed up a string reference that the GUI doesn't recognise.

6.4.2.4.2.9 Failed to open file <filename> for writing/reading

The GUI couldn't open the file specified. Check write permissions.

6.4.2.5 Examples

```

,*****
;8 bit C-BUS read, write to screen and file
,*****
bar    word    0

        register #1, #$10, #1                ;set length of register $10
                                                ;in device 1 to 8 bits
        device 1                             ;select device 1
        fopenw "FileWrite.txt"               ;open file for writing
        copy   *$10, bar                     ;read from C-BUS, store in bar
        disp   "%x", bar                     ;write to screen
        filew  "%x", bar                     ;write to file
        stop                                     ;end of script

,*****
;Copy from one device to another
,*****
bar      buffer 10
index    word    0

        device 1                             ;select device 1
        copy   #0, index
        while  index < #10
            copy   *$b5, bar[index++]        ;read from C-BUS, store in bar[n]
                                                ;inc index
        endwhile
        device 2                             ;select device 2
        copy   #0, index
        while  index < #10
            copy   bar[index++], *$a7        ;read from bar[n] store to C-BUS
                                                ;inc index
        endwhile
        stop                                     ;end of script

,*****
;Write from a file into a C-BUS address, controlled by flags
,*****
counter    word    1
status     word    1
temp       word    1

        fopenr "inputfile.txt", "%04X"      ;tell PC to open file & stream
                                                ;data down
        cls                                  ;clear display window
        copy   #0, counter                   ;set up loop counter
        while  counter < #32                 ;start while loop
            filew temp                        ;read from file buffer
            write temp, *$a7                  ;write to C-BUS
            jsr  read_and_wait                ;wait for status flag
            add  #1, counter                   ;increment counter
            disp "Written %i.", counter       ;write to screen
        endwhile                             ;end of while loop
        dialog "Finished"                    ;message box
        stop                                  ;end of script

read_and_wait
        copy   *$c1, status                  ;read C-BUS address $c1 into variable
        and    #$0010, status                 ;mask off wanted bits
        jmpc   status != #$0010, read_and_wait ;if bit 4 not set, try again.
        return

```

6.4.3 The Function Image™ Load Tab

This tab provides Function Image™ load and activation utilities for *FirmASIC*® products. The operation of this tab is described in the user manuals of the {target} cards to which it applies.

6.4.4 The Serial Memory Programming Tab

This tab provides a serial memory programming facility for *FirmASIC*® {target} cards where a serial EEPROM or flash memory can be used for non-volatile storage of a Function Image™. The operation of this tab is described in the user manuals of the {target} cards to which it applies.

6.5 Evaluation Tests

Details of any {target} cards specific tests are included in the relevant {target} cards user manual.

6.6 TroubleShooting

TBD

7. Performance Specification

7.1 Electrical Performance

7.1.1 Absolute Maximum Ratings

Exceeding these maximum ratings can result in damage to the Evaluation Kit.

	Min.	Max.	Units
Supply (VCCIN - GNDD)	-0.3	5.5	V
Voltage on any connector pin to GNDD	-0.3	3.6	V
Current into or out of VCCIN and GNDD pins	-50	+300	mA
Current into or out of any other connector pin	-20	+20	mA
Storage Temperature	-10	+70	°C
Operating Temperature	+10	+35	°C

7.1.2 Operating Limits

Correct operation of the Evaluation Kit outside these limits is not implied.

	Notes	Min.	Max.	Units
Supply (VCCIN - GNDD)		4.8	5.5	V
Operating Temperature		+10	+35	°C
Xtal Clock Frequency		19.99	20.01	MHz

7.1.3 Operating Characteristics

For the following conditions unless otherwise specified:

Xtal Frequency = 20MHz
VCCIN = 5.0V, Tamb = +25°C.

	Notes	Min.	Typ.	Max.	Units
DC Parameters					
I _{CCIN} (not powersaved)	1, 2		150	250	mA
I/O pins					
Input logic "1" level	3	70%			3.3V
Input logic "0" level	3			30%	3.3V
Input leakage current	3	-100	±1	+100	nA
Output Logic '1' Level @ I _{OH} = 1.6 mA	3	2.8			V
Output Logic '0' Level @ I _{OL} = -1.5 mA	3			0.4	V
C-BUS Interface					
Input logic "1" level	4	2.9		3.6	V
Input logic "0" level	4	-0.3		0.6	V
Input leakage current	4			±10	µA
Output leakage current	4			±10	µA
Output Logic '1' Level @ I _{OH} = 1.0 mA	4	2.4			V
Output Logic '0' Level @ I _{OL} = -4 mA	4			0.45	V
Input/Output Capacitance	4			10	pF
C-BUS clock frequency			5.0		MHz

Notes:

1. Not including any current drawn from the output pins by external circuitry.
2. No Target Card connected.
3. RS232/CBUS, BOOTEN1, BOOTEN2, GPIO0, GPIO1, GPIO2, GPIO3, IO0, IO1, IO2, IO3, IO4, IO5, IO6, IO7 and IO3 pins.
4. CSN1, CSN2, IRQN1, IRQN2, RDATA, CDATA and SCLKCBUS pins.

CML does not assume any responsibility for the use of any circuitry described. No IPR or circuit patent licences are implied. CML reserves the right at any time without notice to change the said circuitry and any part of this product specification. Evaluation kits and demonstration boards are supplied for the sole purpose of demonstrating the operation of CML products and are supplied without warranty. They are intended for use in a laboratory environment only and are not for re-sale, end-use or incorporation into other equipments. Operation of these kits and boards outside a laboratory environment is not permitted within the European Community. All software/firmware is supplied "as is" and is without warranty. It forms part of the product supplied and is licensed for use only with this product, for the purpose of demonstrating the operation of CML products. Whilst all reasonable efforts are made to ensure that software/firmware contained in this product is virus free, CML accepts no responsibility whatsoever for any contamination which results from using this product and the onus for checking that the software/firmware is virus free is placed on the purchaser of this evaluation kit or development board.

 CML Microcircuits (UK) Ltd COMMUNICATION SEMICONDUCTORS	 CML Microcircuits (USA) Inc. COMMUNICATION SEMICONDUCTORS	 CML Microcircuits (Singapore) Pte Ltd COMMUNICATION SEMICONDUCTORS SingaporeChina	
Tel: +44 (0)1621 875500 Fax: +44 (0)1621 875600 Sales: sales@cmlmicro.com Tech Support: techsupport@cmlmicro.com	Tel: +1 336 744 5050 800 638 5577 Fax: +1 336 744 5054 Sales: us.sales@cmlmicro.com Tech Support: us.techsupport@cmlmicro.com	Tel: +65 67450426 Fax: +65 67452917 Sales: sg.sales@cmlmicro.com Tech Support: sg.techsupport@cmlmicro.com	Tel: +86 21 6317 4107 +86 21 6317 8916 Fax: +86 21 6317 0243 Sales: cn.sales@cmlmicro.com.cn Tech Support: sg.techsupport@cmlmicro.com
- www.cmlmicro.com -			