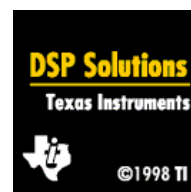




HUNT ENGINEERING
Chestnut Court, Burton Row,
Brent Knoll, Somerset, TA9 4BP, UK
Tel: (+44) (0)1278 760188,
Fax: (+44) (0)1278 760199,
Email: sales@hunteng.demon.co.uk
URL: <http://www.hunteng.co.uk>



We are a
committed
member of the
Texas Instruments
3rd party
programme

For Sales and Support in North America Please Contact Our Strategic Partner:

Traquair Data Systems Inc, 114 Sheldon Road, Ithaca, NY 14850 USA

Tel 607 266 6000, Fax 607 266 8221

Email Traquair@traquair.com, URL <http://www.traquair.com>

For Sales and Support in Other Areas Please Contact Your Local Reseller.

HEPC2104

'C44 PC/104 Card

USER MANUAL

Hardware Rev B
Document Rev D
R.Williams 21/6/99

COPYRIGHT

This documentation and the product it is supplied with are Copyright HUNT ENGINEERING 1999. All rights reserved. HUNT ENGINEERING maintains a policy of continual product development and hence reserves the right to change product specification without prior warning.

WARRANTIES LIABILITY and INDEMNITIES

HUNT ENGINEERING warrants the hardware to be free from defects in the material and workmanship for 12 months from the date of purchase. Product returned under the terms of the warranty must be returned carriage paid to the main offices of HUNT ENGINEERING situated at BRENT KNOLL Somerset UK, the product will be repaired or replaced at the discretion of HUNT ENGINEERING.

Exclusions - If HUNT ENGINEERING decides that there is any evidence of electrical or mechanical abuse to the hardware, then the customer shall have no recourse to HUNT ENGINEERING or its agents. In such circumstances HUNT ENGINEERING may at its discretion offer to repair the hardware and charge for that repair.

Limitations of Liability - HUNT ENGINEERING makes no warranty as to the fitness of the product for any particular purpose. In no event shall HUNT ENGINEERING'S liability related to the product exceed the purchase fee actually paid by you for the product. Neither HUNT ENGINEERING nor its suppliers shall in any event be liable for any indirect, consequential or financial damages caused by the delivery, use or performance of this product.

Because some states do not allow the exclusion or limitation of incidental or consequential damages or limitation on how long an implied warranty lasts, the above limitations may not apply to you.

TECHNICAL SUPPORT

Technical support for HUNT ENGINEERING products should first be obtained contacting your local supplier, if you are unsure of details please refer to www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.demon.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.

TABLE OF CONTENTS

INTRODUCTION.....	5
GETTING STARTED.....	7
PHYSICAL LOCATION OF ITEMS ON THE HEPC2104.....	8
GENERAL DESCRIPTION OF INTERFACES.....	9
THE HOST COMPORT INTERFACE.....	10
HEPC2104 I/O MAP.....	10
FIFO FLAGS REGISTER.....	11
<i>Using the FIFO Flags.....</i>	<i>11</i>
INTERRUPTS.....	12
CONTROL REGISTER.....	13
THE POWER DOWN FEATURE.....	14
TIM-40 RESET.....	15
TIM-40 CONFIG.....	16
THE CONTROL HEADERS.....	17
HEPC2104 SYSTEM CONFIGURATION EXAMPLES.....	18
MASTER BOARD (DEFAULT SETTING).....	18
SLAVE BOARD (SINGLE HOST CONNECTION SYSTEM).....	19
SLAVE BOARD (MULTI HOST CONNECTION SYSTEM).....	20
'C44 MEMORY CONFIGURATION.....	22
THE GLOBAL FLASH MEMORY OPTION.....	23
FLASH BOOTING.....	23
ID ROM.....	25
COMPORT CONNECTIVITY.....	27
I/OS AND TIMERS.....	28
LEDS.....	29
HARDWARE DETAILS.....	30
GENERAL INFORMATION.....	30
COMPORT CONNECTORS.....	30
THE 'RES SEL' HEADER.....	32
CONTROL CONNECTORS.....	32
<i>Interconnecting Motherboards.....</i>	<i>33</i>
THE IIOFs HEADER.....	33
THE TIMER 1 HEADER ('TCK').....	34
THE 'PWR DWN' HEADER.....	34
JTAG CONNECTOR.....	35
POWER CONNECTOR (OPTIONAL).....	35
PHYSICAL DIMENSIONS OF THE HEPC2104.....	37
POWER REQUIREMENTS OF THE HEPC2104.....	38
SOFTWARE.....	39
DEVELOPMENT TOOLS.....	39

3L Parallel C.....	39
HUNT ENGINEERING Server/Loader.....	39
Go DSP Code Composer.....	39
Application Programming Interface (API)	40
SOFTWARE REQUIREMENTS OF THE HEPC2104	40
The Host Interface	40
Configuring for use with 3L Parallel C.....	41
Configuring for use with the Server/Loader	41
Using the JTAG Debugger with other Tools.....	42
Using the TI Compiler/Assembler	42
The C Source Debugger.....	45
EXAMPLE SOFTWARE FOR PC/104 HOST SOFTWARE DEVELOPMENT	46
EXAMPLE SOFTWARE FOR A HEPC2104 WITH GLOBAL FLASH	50
Using the 3L ROM Booting Kit.....	51
Programming the FLASH	52
CE MARKING.....	56
TECHNICAL SUPPORT	57

The HEPC2104 is a 16-bit PC/104 format board that provides an interface to a TIM-40 'C4x communication port (or Comport). The PC/104 module also has an on-board TMS320C44 DSP processor that can be used with a TIM-40 system. There are four 'C44 Comports connected to the DSP, one of which can be used as a dedicated connection to the host Comport interface of the PC/104 module.

The TMS320C44 (or 'C44) is a version of the popular TMS320C40 from Texas Instruments. It is a lower cost version in a 'Plastic Quad Flat Pack' with less pins. This is made possible by only connecting four of the Comports out of the package, and only connecting 24 bits of each of the Local and Global address busses. Thus it is software compatible with the 'C40, and provides the same performance, but in a cheaper package that consumes less PCB real estate. The 'C44 has a peak performance of 60 MFlops and 330 Mops and both memory busses are capable of 120 MBytes/sec. The Comports are rated at 20 MBytes/sec.

Used with the 'C44 processor and SRAM memory, the host Comport interface of the HEPC2104 provides a simple-to-use connection to the DSP and/or other TIM-40 hardware. However by not fitting the processor and associated logic, the HEPC2104 can be configured as a host Comport interface only, enabling it to be used to host a separate TIM-40 system. Thus, multiple 'C4x networks may be made with other HEPC2104 modules or any of the HUNT ENGINEERING TIM-40 modules and motherboards.

When fitted with the 'C44 processor, the module is available with varying sizes of zero wait state SRAM memory fitted on both the Local and Global Bus of the 'C44. The HEPC2104 is also alternatively available with FLASH memory fitted on the Global Bus instead of SRAM. With FLASH memory fitted the HEPC2104 can boot with an application previously stored in the FLASH. As such the module can be used for embedded system applications where self booting is required.

When the HEPC2104 is used with the FLASH global memory option the board provides a boot jumper that defines whether on reset the processor boots from a Comport (for example the host Comport interface) or from the global bus.

The HEPC2104 provides a standard 14-way header for connection to an external JTAG Master such as an XDS-510 or a HUNT ENGINEERING HEPC2E, HEV40-4 or HESB40. This allows debugging of the on-board 'C44.

The HEPC2104 supports a global reset scheme for the on-board 'C44 (and any other connected 'C4x hardware). It can be used as either the master of the reset using the PC/104 interface, or as a slave with the 'C44 and host interface being reset from another board.

The 'C44, its SRAM and associated interface circuitry may be powered down under software control.

The HEPC2104 provides an interrupt mask register that can be used to configure the board to drive one of the interrupt lines of the PC/104 bus.

Although the HEPC2104 is not a TIM-40 module, much of the TIM-40 specification is adhered to, in order to be compatible with other TIM-40 hardware and software. As such, the 'C44 and host Comport interface can be directly connected and used with other TIM-40 products. However, please note that the HEPC2104 Revision C board replaces the older

Revision B board, and is not software or hardware compatible. The board also uses the HUNT ENGINEERING 2nd generation Comport scheme. Therefore when connecting to other TIM-40 hardware please ensure the compatibility of the boards connected.

The HEPC2104 is a module that plugs into a PC/104 stack. The HEPC2104 comes configured ready to be fitted to your machine as a single DSP board. If you wish to reconfigure the board for use as a secondary board in a system see later sections in this manual.

The default jumper settings for the HEPC2104 are shown with the use of black boxes on the appropriate headers in the diagram in the following section.

The HEPC2104 can be set up with jumpers to respond to 16-bit I/O accesses, note: PC/104 bus memory accesses are not supported by the board. The default setting is that it uses I/O space starting at a base address of 0x200. Alternatively, the base address can be set to 0x160 or 0x300. The HEPC2104 uses a PC/104 byte address range of 0 to 6 (inclusive), starting from the base address selected. If the I/O address range clashes with other modules or peripherals (or is different from that required by the software) then it will need to be relocated by changing the HEPC2104 base address jumpers, refer to the relevant sections in this manual to do this.

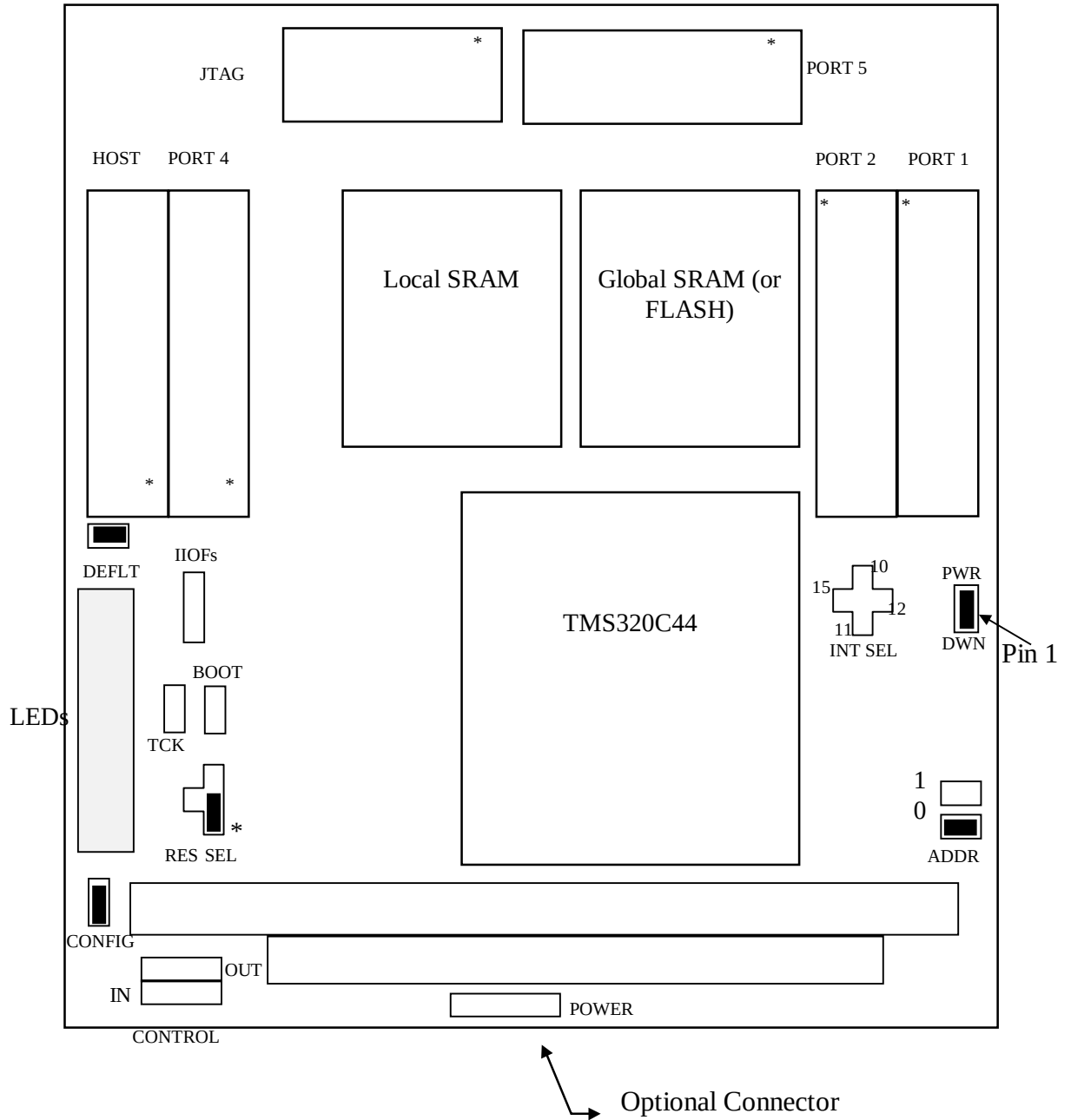
The default setting for interrupts is disabled. The possible interrupts that can be used are IRQ10, IRQ11, IRQ12 and IRQ15. To enable interrupts a jumper is required on the interrupt select header.

Before installing the HEPC2104, first remove all power from the PC/104 machine. Gently connect the HEPC2104 onto the top or bottom of the PC/104 stack ensuring that the bus connectors are mated correctly and that the spacers are fitted where appropriate. Do NOT use excessive force.

Re-apply power to the machine and ensure the machine boots normally. If it does not, this probably means that there is an address or interrupt clash between the HEPC2104 and the PC/104 system. Re-check that the address range you have selected with the address jumpers does not clash with existing peripherals. Also check the interrupt selected does not clash with other hardware.

If the machine boots properly then follow the installation and user guide for the software that you have chosen, but remember that if you have problems when the software is used it might still be due to address or interrupt clashes. It is always a good idea to use any confidence checks that are supplied with the software before trying your own as then you can eliminate software problems as the cause.

Physical Location of Items on the HEPC2104



The * indicates pin 1

The Black jumpers indicate fitted in the default ship state

General Description of Interfaces

The main interface on the HEPC2104 is the 16-bit I/O interface between the PC/104 bus and the host Comport. The host Comport interface provides a path for both Comport data and control between the PC/104 host and a TIM-40 system. The Comport data path contains a bi-directional FIFO that buffers data between the 16-bit PC/104 bus and the byte wide host Comport.

In addition to the Comport data path, there are also several control registers that are interfaced to the PC/104 bus. The first register is the FIFO status and interrupt register. By reading this register it is possible to ascertain the state of the host Comport FIFO flags. By writing to the register interrupt causes can be set for three FIFO flag conditions. As such this register provides a easy method to control the flow of Comport data.

The second register (the 'control register'), allows the host interface to set the control signals of the HEPC2104 host interface. This set of signals comprises of a global reset output, a power-down control output, and a FIFO endian switch. The register enables the HEPC2104 to become the master of the TIM-40 system wide Reset signal, it enables a large portion of the module to be powered down when required, and it enables the setting of the endian transfer of data through the Comport FIFO.

The power down feature of the HEPC2104 is interfaced to a 2-pin 'PWR DWN' header. This header allows the host interface to control the power enable of this board and others. Alternatively using this header the power enable for this board can be driven from another board. As a result, it is possible to connect many HEPC2104 boards together where the power down feature of all boards is controlled by one master board only.

The next interface of the HEPC2104 comprises of the 'CONTROL OUT' and 'CONTROL IN' headers. These two headers allow the HEPC2104 to be connected as part of a larger TIM-40 system. The HEPC2104 may drive the global system reset for all the boards in a system, or alternatively it can be the slave of another board that controls the reset.

The next interface is the JTAG header. This allows the JTAG port of the 'C44 to be exercised by the relevant hardware and software tools. The JTAG slave connections are provided on a standard 14-way header. As such a standard XDS-510 type cable can be connected and the 'C44 can become the slave of an XDS-510 type debugging system.

To enable the user to configure their own system topology there are four 'C44 Comport connectors in addition to the host Comport connector. These allow for connections to other 'C4x Comports on other boards if a multi-board system is required. The HEPC2104 uses the 2nd generation HUNT ENGINEERING Comport buffering scheme, the advantages of which include reduced power consumption and higher bandwidth. This buffering scheme is not compatible with the earlier schemes used by HUNT ENGINEERING or with schemes used by other vendors.

The Host Comport Interface

The PC/104 host can transfer data to and from the 'Host' Comport by reading and writing the host port FIFO. This bi-directional FIFO interfaces the 16-bit PC/104 bus to the byte wide Comport. It must be remembered that 'C4x Comport communication is always in 32 bit words, so any software must always perform reads and writes in blocks of four bytes. This is done by performing two 16-bit accesses to the FIFO for each read or write.

There is a single location for the read data (from Comport to host) and the write data (from host to Comport). By reading the host FIFO status and interrupt register location the state of the FIFO flags can be ascertained. This enables the communicating software to ensure the transfer of data can be completed correctly. As writing to a full FIFO will cause the data to be lost and reading from an empty FIFO will provide invalid data, it can be seen that the correct use of the FIFO flags is important.

An interrupt can be enabled on any one of three possible FIFO flag events. This allows communicating software to wait for an interrupt to occur, showing that a FIFO transfer can proceed.

HEPC2104 I/O Map

The base address in I/O space can be set using the 'ADDR' jumpers located next to the PC/104 connectors. With the jumpers correctly fitted the HEPC2104 will respond to 16-bit accesses to the locations within the selected byte address range. The base address options are detailed below. A shaded box represents a jumper fitted.

1 ☐	1 ☐	1 ☒	1 ☒
0 ☐	0 ☒	0 ☐	0 ☒
0x160	0x200	0x300	De-selected
<u>Addr 0</u>	<u>Addr 1</u>	<u>I/O Base Address</u>	<u>I/O Address Range</u>
not fitted	not fitted	0x160	0x160 - 0x166
fitted	not fitted	0x200	0x200 - 0x206
not fitted	fitted	0x300	0x300 - 0x306
fitted	fitted	not in host address map	

The location of the FIFO and registers are organised as follows, offset from the base address. To access these locations a 16-bit I/O read or a 16-bit I/O write is required.

<u>Byte Offset</u>	<u>Read/Write</u>	<u>Function</u>
0x0	R/W	Host Comport FIFO
0x4	R	FIFO Flags
0x4	W	Interrupt Mask
0x6	W	Control Register

FIFO Flags Register

The FIFO Flags Register is a single 16-bit register that contains all of the flags for the host Comport FIFO. It is read only, at byte offset 4 from the selected base address. All of the flags are active low. That is, when any of the bits of the flag's register are read low, then that flag condition is true.

<u>Register Bit</u>	<u>FIFO Flag</u>
0	PC/104 to Comport FIFO Full Flag
1	PC/104 to Comport FIFO Almost Full Flag
2	Comport to PC/104 FIFO Empty Flag
3	Comport to PC/104 FIFO Almost Empty Flag
4	Comport to PC/104 FIFO Full Flag
5	Comport to PC/104 FIFO Almost Full Flag
6	PC/104 to Comport FIFO Empty Flag
7	PC/104 to Comport FIFO Almost Empty Flag
8 - 15	Undefined

The HEPC2104 host FIFO comprises of two independent sixty four word FIFOs, where each word is 32-bits in size (one 'C4x word). As such two 16-bit accesses to the FIFO location when reading or writing involves one FIFO position.

Using the FIFO Flags

This section provides a description of what each of the FIFO flags represents when set low or high, and how it is recommended the flags are used when transferring Comport data.

1. PC/104 to Comport FIFO Full Flag: when not true (register bit 0 set high) this flag indicates that there is space to write a full 'C4x word to the FIFO, with two 16-bit writes to the HEPC2104 Comport FIFO.
2. PC/104 to Comport FIFO Almost Full Flag: when not true (register bit 1 set high) this flag indicates that there is space to write four 'C4x words, with 8 16-bit writes to the HEPC2104 Comport FIFO.
3. Comport to PC/104 FIFO Empty Flag: when not true (register bit 2 set high) this flag indicates that there is one 'C4x word to read, with two 16-bit reads from the HEPC2104 Comport FIFO.
4. The Comport to PC/104 FIFO Almost Empty Flag: when not true (register bit 3 set high) this flag indicates that there is at least four 'C4x words to read, with 8 16-bit reads from the HEPC2104 Comport FIFO.
5. The Comport to PC/104 FIFO Full Flag: when true (register bit 4 set low) this flag indicates that there is sixty-four 'C4x words to read, with 128 16-bit reads from the HEPC2104 Comport FIFO.

NOTE: The Comport interface will aim to prevent this situation from occurring in order to avoid a potential deadlock situation with the connected 'C4x (refer to the section on Software Requirements). This means that the full flag true condition should not be relied upon for reading Comport data from the host interface. Instead the Comport to PC/104

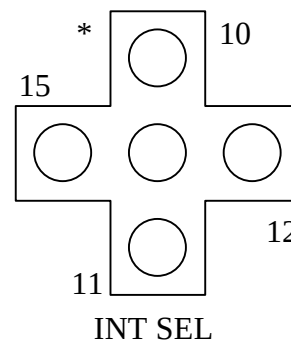
FIFO Almost Full Flag should be used.

6. The Comport to PC/104 FIFO Almost Full Flag: when true (register bit 5 set low) this flag indicates that there is at least sixty 'C4x words to read, with 120 16-bit reads from the HEPC2104 Comport FIFO.
7. The PC/104 to Comport FIFO Empty Flag: when true (register bit 6 set low) this flag indicates that there is space to write sixty-four 'C4x words, with 128 16-bit writes to the HEPC2104 Comport FIFO.
8. The PC/104 to Comport FIFO Almost Empty Flag: when true (register bit 7 set low) this flag indicates that there is space to write sixty 'C4x words, with 120 16-bit writes to the HEPC2104 Comport FIFO.
9. When transferring data to and from the HEPC2104 the use of block transfers where possible will result in a higher performance than many single word accesses. It should be noted that using FIFO flags that indicate a block of data can be transferred will be more efficient, requiring fewer bus cycles per 'C4x word transferred, hence maximising performance.

Interrupts

Although it is entirely possible to use the HEPC2104 without using interrupts, three possible interrupt causes are provided for users who wish to use this feature.

After reset, all interrupts are disabled, and therefore will not interfere with the users system. It is possible to select between four PC/104 interrupts, IRQ10, IRQ11, IRQ12, and IRQ15. To connect the HEPC2104 interrupt output to one of these interrupt lines a jumper must be fitted to the 'INT SEL' header between the centre pin and the interrupt required.



The interrupt causes are taken from the FIFO flags so all of the possible causes can be obtained by reading the FIFO Flags register. The Interrupt Mask register is write only and is initialised to zero on reset (that is, on assertion of the ISA bus signal 'RESETDRV'). The Interrupt Mask register is a 16-bit write only register at byte offset 4.

<u>Register Bit</u>	<u>Interrupt Enable</u>
0	PC/104 to Comport No Longer Almost Full
1	Comport to PC/104 Has Become Almost Full
2	Comport to PC/104 No Longer Empty

The above register bits must be set high to enable the corresponding interrupt.

If there are multiple causes for an interrupt and only one is serviced then the interrupt output remains asserted. If level sensitive interrupt mode is selected in the PC/104 system, re-enabling the interrupt will cause a new interrupt. If edge triggered interrupts are used then the interrupt service routine should disable all HEPC2104 interrupts and then re-enable any that were previously enabled and have not been serviced.

The intended use of interrupts is as follows:

1. 'No Longer Almost Full': When a write to the 'C4x Comport from the PC/104 has been blocked because the FIFO has become almost full, enabling and receiving this interrupt gives an indication that some room has become available for the write to continue. Please note that by the time the interrupt service routine has been called there might be more room available, so it is a good idea for the ISR to read the FIFO flags to establish the real amount of FIFO space.
2. 'No Longer Empty': When a read from the 'C4x Comport by the PC/104 has been blocked by the lack of data to read, enabling and receiving this interrupt will give an indication when the first word arrives from the 'C4x. In some cases it would be better to use case 3.
3. 'Has Become Almost Full': If a large read from the Comport is outstanding it would be more efficient to read a large block of data from the FIFO. In this case this interrupt gives an indication that at least 60 'C4x words of data can be read from the FIFO.

Control Register

The control register is a 16-bit bit write only register at byte offset 6. The function of this register is shown below.

<u>Register Bit</u>	<u>Control Function</u>
0	Host Reset
1	Host FIFO Endian Control
2	Power Down Enable
3 - 15	Undefined

The host reset is an active low reset. This reset is buffered onto pin 2 of the 'RES SEL' header and can be used to drive the system wide reset by fitting a jumper to this header. To assert the host reset, a zero must be written to bit 0 of the control register, and then to de-assert a one must be written to bit 0 of the control register. Note, whenever the ISA bus reset signal, 'RESETDRV' is asserted the host reset will also be asserted, but removed when 'RESETDRV' is removed, making the 'C44 ready for use with debugger tools if required.

The host FIFO can perform an endian swap on the Comport data. For little endian operation the host FIFO endian control bit must be written as zero. For big endian operation the host FIFO control bit must be written as one. The following table shows the relationship between the endian setting and the transfer of bytes between the PC/104 bus and the host Comport. Note, for each full 32-bit word transfer, the first bus transfer is a 16-bit transfer of bits D0-D15 and the second bus transfer is bits D16-D31.

<u>Comport Byte Order</u>	<u>Endian Bit = 0</u>	<u>Endian Bit = 1</u>
first	D0-7	D24-31
second	D8-15	D16-23
third	D16-23	D8-15
fourth	D24-31	D0-7

The Power Down Feature

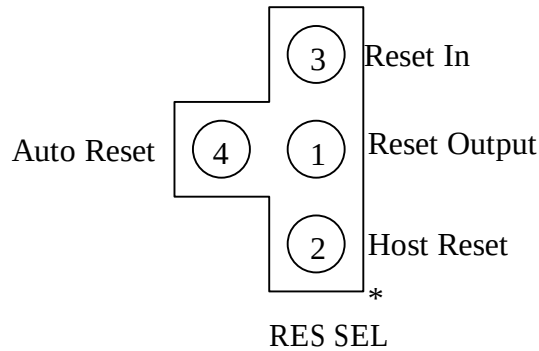
When operation of the HEPC2104 is not required, the majority of the HEPC2104 hardware can be powered down independently from the host interface control logic to significantly reduce power consumption. The 'power-down' logic (that is, the HEPC2104 circuitry that can be powered down) includes the host comport FIFO, the host Comport connection, and the 'C44 and memory.

The power down enable bit of the control register is buffered onto the 'PowerEn Out' signal on pin 1 of the 'PWR DWN' header. This header can be used to drive the 'PowerEn In' signal on pin 2 of the 'PWR DWN' header by simply fitting a jumper to the header. Alternatively by connecting cables between the 'PWR DWN' of multiple boards, the first 'PowerEn Out' signal can be used to drive the 'PowerEn In' signal of all boards connected. In this manner the power down enable bit of the first board can be used to power down all boards.

NOTE if the power down feature is used in a system comprising of multiple boards that have interconnected Comports, then all Comport connected boards must have the power down enable signal connected. This is to ensure that if the system is powered down, both ends of any Comport are powered down.

To turn on all power-down logic the power down enable bit must be written low. To turn off all power-down logic the power down enable bit must be set high. On reset of the PC/104 system (that is, on assertion of the ISA bus signal 'RESETDRV') the power down enable is set to zero.

The 'C44 Reset on the HEPC2104 is buffered from the centre pin (pin 1) of the 'RES SEL' header. By fitting a jumper between the centre pin of the 'RES SEL' header and pin 2, pin 3 or pin 4 any one of the following three reset sources can be selected.



Host Reset: By fitting a jumper between pin 1 and pin 2, the host reset can be used to drive the system wide TIM-40 reset. The host reset is controlled from the host interface Control Register.

Reset In: By fitting a jumper between pin 1 and pin 3, the 'Reset In' signal from pin 1 of the 'Control In' header can be used to drive the 'C44 reset on the HEPC2104.

Auto Reset: By fitting a jumper between pin 1 and pin 4, the 'Auto Reset' feature can be used to reset the board for a short time after power up. This power up reset takes effect on powering up the 'power-down' logic on the HEPC2104. This feature is intended for use with a HEPC2104 fitted with FLASH memory on the global bus. The auto-reset enables the module to boot from an application stored in the global FLASH memory. This feature enables the HEPC2104 to be used as part of a self booting system. (Refer to the FLASH memory section of this manual.)

The Reset Output taken from the centre of the 'RES SEL' header is buffered and driven onto the 'Reset Out' pin of the 'CONTROL OUT' header. In this manner, the HEPC2104 reset output can be used to drive the reset in of other connected boards.

The TIM-40 Config signal is a system wide signal that enables ‘dangerous’ portions of hardware to be held in a safe state until all processors have been booted. When all processors in the system become booted and software control can begin, the system wide Config line can be released and these ‘dangerous’ elements can safely begin operation.

The Config signal is provided on the control headers to facilitate the connection of multiple boards all on the same Config line.

The TIM-40 Config line is an open-collector line that becomes asserted when the TIM-40 reset is applied. After reset the 'C44 on the HEPC2104 will drive the Config signal low by using a pull up resistor to switch on a transistor. This transistor can be switched off, thereby removing the drive on the Config signal, by making the 'C44 drive the IIOF3 pin as a logic low output.

Therefore in order to release the Config signal after a reset a software routine must be called to set the IIOF3 pin to be a logic low output. By setting the 'C4x IIF register bit 12 low and by setting bit 13 high, the IIOF3 pin becomes an output. Then by setting the 'C4x IIF register bit 14 low, the output is driven low. For more details refer to the 'C4x User Guide section on the IIF register.

A suitable piece of 'C4x assembly code to release the Config line is shown below. This code must be executed by all 'C4x processors in a system.

```
push R1          * preserve the value of R1
ldi  IIF,R0      * read the IIF register
ldi  0f000h,R1   *
not  R1          * get ffff0fff in R1
and  R1,R0       * mask the old IIF register value
or   2000h,R0    * set IIOF3 as output I/O pin, and low
ldi  R0,IIF      * put value in IIF register
pop  R1          * restore R1
```

The Config signal requires a single pull-up resistor to be fitted in any system configuration. To connect a pull up resistor on the HEPC2104, a jumper must be fitted to the ‘Config’ header.

NOTE: only one ‘Config’ jumper must be fitted in a system.

A good method is to fit the Config jumper only on the board that is the master of reset, as there must also only be one RESET master.

The HEPC2104 provides a 'CONTROL OUT' header and a 'CONTROL IN' header, situated near the PC/104 connectors. These headers allow multiple boards to be connected enabling the use of the global signals TIM-40 Reset and TIM-40 Config. By connecting the 'CONTROL OUT' header to the 'CONTROL IN' of another HUNT ENGINEERING board, the HEPC2104 can be used to control the reset of the system. The reset driven out on this connection is selected by the 'RES SEL' header (refer to the section on TIM-40 Reset). Also by connecting the Control headers of multiple boards, this will join the Config signal to all the boards in the system (refer to the section on TIM-40 Config).

By connecting the 'CONTROL IN' header to the 'CONTROL OUT' of another HUNT ENGINEERING board, the HEPC2104 can be used as a slave of reset. In this configuration, the 'RES SEL' header would require a jumper between pins 1 and 3 to connect the 'Reset In' from the 'CONTROL IN' header to the boards reset signal. In this way the 'C44 and host Comport of the HEPC2104 can be reset from the other HUNT ENGINEERING board.

With the HEPC2104 connected as a slave of reset, the 'CONTROL OUT' header can be connected to the 'CONTROL IN' of the next board to propagate the system reset to other boards.

As such by correctly cabling together the control headers of many boards using standard HUNT ENGINEERING Reset Cables, the required system control topology can be achieved.

HEPC2104 System Configuration Examples

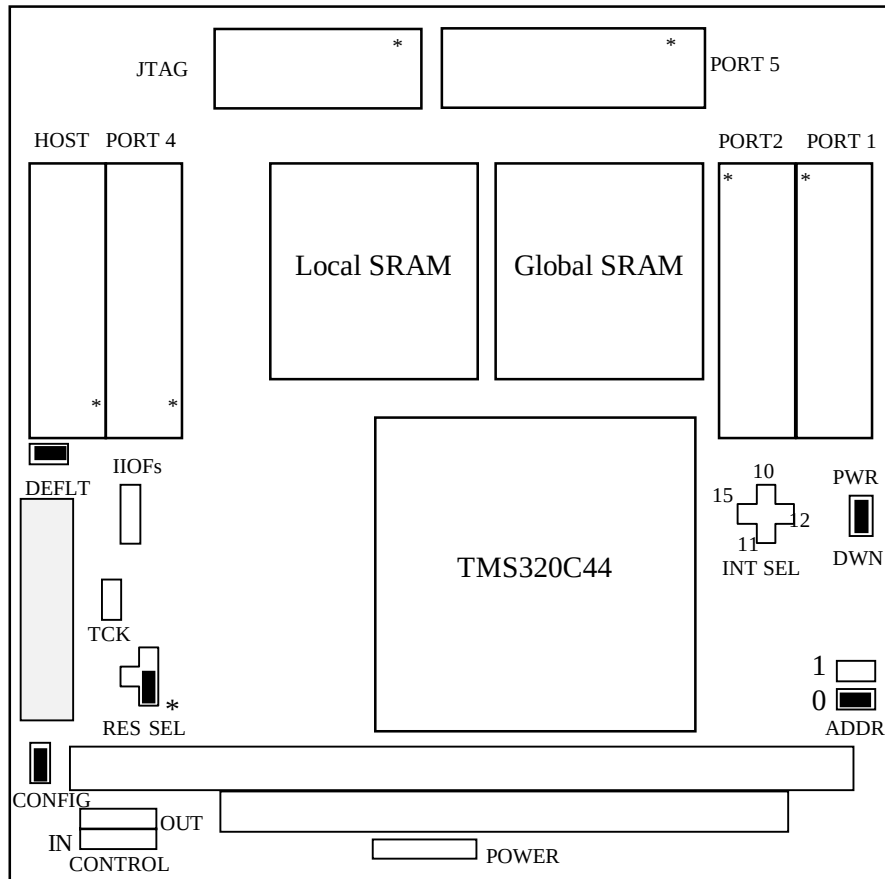
This section provides some examples of how the HEPC2104 should be set up for typical system configurations. Examples are given for a single HEPC2104 system, for a HEPC2104 hosted system, and for a HEPC2104 slave system. These examples are by no means exhaustive of all the possible uses of the HEPC2104, but they aim to highlight the issues common to all system configurations.

Master Board (Default Setting)

If the HEPC2104 is to be used as a single 'C44 processor system, or the first in a multi-board system, it should be configured as a master board. In this configuration, the host Comport interface of the HEPC2104 is connected directly to Comport 4 of the 'C44 using the 'default' connection, which requires no Comport cabling to be fitted.

For this configuration, the following jumpers would be fitted.

1. 'ADDR 0' and 'ADDR 1' Jumpers: The 'ADDR' jumpers must be fitted to select the correct base address to be used by the host software.
2. 'INT SEL' Jumper: If interrupts are required, a jumper must be fitted to select the correct interrupt line.
3. 'PWR DWN' Jumper: A jumper must be fitted to the 'PWR DWN' header to ensure the host interface is driving the power down enable input for the power down control circuitry.
4. 'RES SEL' Jumper: A jumper must be fitted between pins 1 and 2 of the 'RES SEL' header to ensure the host interface controls the reset of the system.
5. 'DEFLT' Jumper: A jumper must be fitted to the 'DEFLT' default Comport connection header. With this jumper fitted, there must not be a Comport cable connection to the 'HOST' header or the 'PORT 4' header. This default connection will directly connect the host comport interface to Comport 4 of the 'C44. Note, if both a Comport cable is fitted and the 'DEFLT' jumper is fitted the 'DEFLT' error LED will illuminate (refer to the section on LEDs).



Example master board, (Host I/F @ 0x200 & no interrupt)

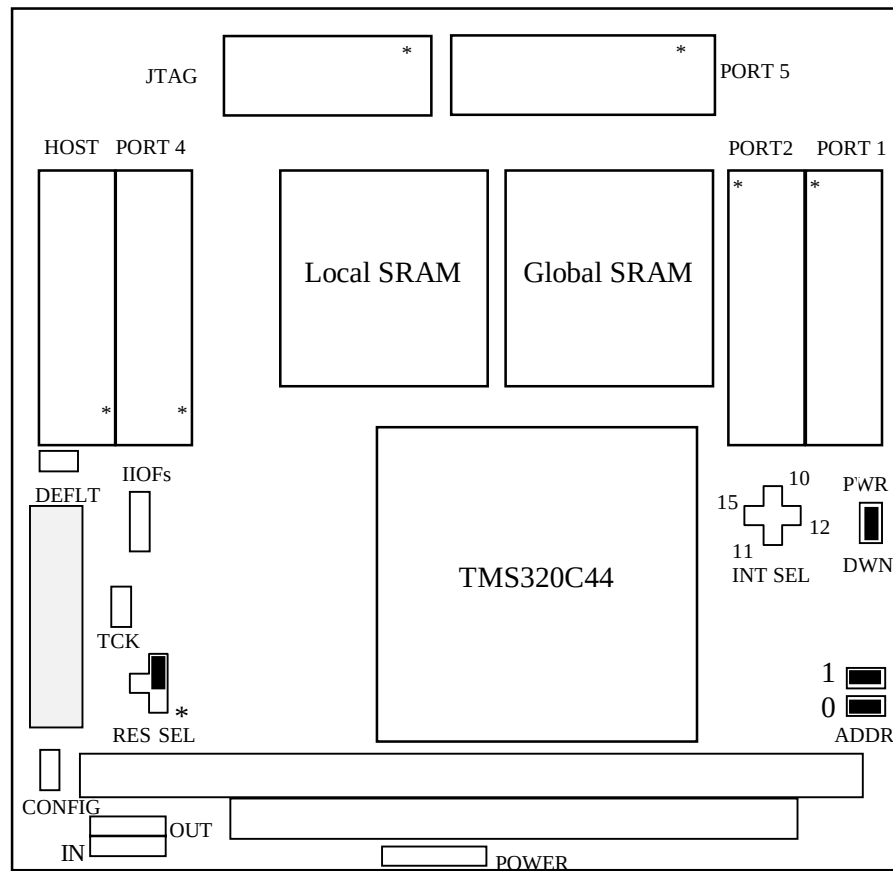
Slave Board (Single Host Connection System)

The HEPC2104 can be used as a slave of another TIM-40 board or of another HEPC2104. The TIM-40 system will include the 'C44 processor. In this configuration, the host Comport interface of the HEPC2104 is not used.

For this configuration, the following jumpers would be fitted.

1. 'ADDR 0' and 'ADDR 1' Jumpers: If this HEPC2104 is a slave to another HEPC2104, the base address jumpers must both be fitted to disable the host interface.
2. 'PWR DWN' Jumper: If this HEPC2104 is a slave to another HEPC2104, a power control cable should be fitted to the 'PWR DWN' header of the slave HEPC2104. This will ensure the host interface of the other board is driving the power down enable input for the power down circuitry controlling the logic on this HEPC2104. Alternatively, if the master board in the system is not a HEPC2104, then a jumper should be fitted to the 'PWR DWN' header.
3. 'RES SEL' Jumper: A jumper must be fitted between pins 1 and 3 of the 'RES SEL' header to ensure the host interface and 'C44 is reset by the 'Reset In' signal on the 'Control In' header.
4. 'Control' Cables: To connect the system Reset signal and Config signal, a standard HUNT ENGINEERING reset cable (or a similar compatible cable) must be connected between the 'CONTROL IN' header of the slave HEPC2104 and the 'CONTROL OUT' header of another TIM-40 board in the system.

5. 'DEFLT' Jumper: A jumper should not be fitted to the 'DEFLT' default Comport connection header.



Example slave board, (no host I/F connection & no power down control)

Slave Board (Multi Host Connection System)

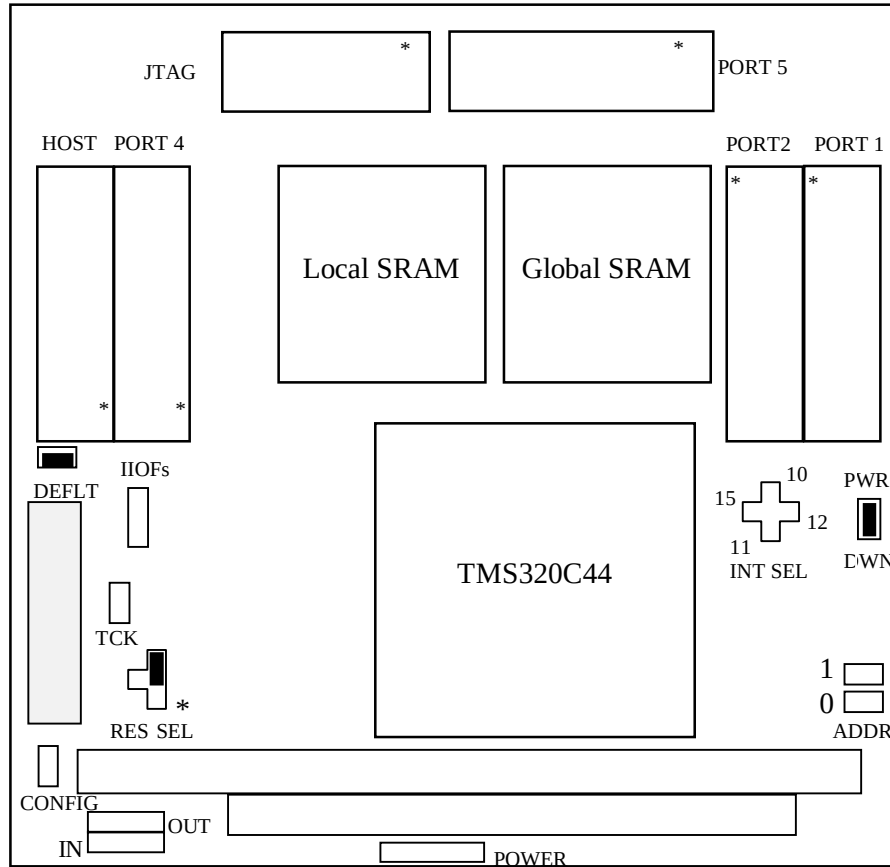
The HEPC2104 can be used as a slave of another TIM-40 board or of another HEPC2104, while still using the 'HOST' comport connection. The TIM-40 system will include the 'C44 processor. In this configuration, the host Comport interface of the HEPC2104 can be used to transfer Comport data, but the Control Register features of the slave HEPC2104 must not be used.

For this configuration, the following jumpers would be fitted.

1. 'ADDR 0' and 'ADDR 1' Jumpers: If this HEPC2104 is a slave to another HEPC2104, the base address jumpers must be fitted to avoid an address clash if the host interface is to be used to transfer Comport data.
2. 'PWR DWN' Jumper: If this HEPC2104 is a slave to another HEPC2104, a power control cable should be fitted to the 'PWR DWN' header of the slave HEPC2104. This will ensure the host interface of the other board is driving the power down enable input for the power down circuitry controlling the logic on this HEPC2104. Alternatively, if the master board in the system is not a HEPC2104, then a jumper should be fitted to the 'PWR DWN' header.
3. 'RES SEL' Jumper: A jumper must be fitted between pins 1 and 3 of the 'RES SEL' header to ensure the host interface and 'C44 is reset by the 'Reset In' signal on the

‘Control In’ header.

4. ‘Control’ Cables: To connect the system Reset signal and Config signal, a standard HUNT ENGINEERING reset cable (or a similar compatible cable) must be connected between the ‘CONTROL IN’ header of the slave HEPC2104 and the ‘CONTROL OUT’ header of another TIM-40 board in the system.
5. ‘DEFLT’ Jumper: A jumper can be fitted to the ‘DEFLT’ default Comport connection header, or another connection made to the ‘HOST’ Comport.



Example slave board, (host I/F connection @ 0x160, no interrupts & no power down control)

'C44 Memory Configuration

The 'C4x has two external memory busses, called the 'Local Bus' and the 'Global Bus'. These are addressed from software on the 'C4x and differentiated by the top bit of the 32 bit address. The 'C4x can access concurrently on both busses.

Each memory bus has 2 independent sets of memory strobes (strobe 0 and strobe 1). Strobe 0 accesses occur at the bottom of the address space of the bus, and Strobe 1 accesses occur at the top of the address space. The point at which they change can be selected by writing to the Bus Control Registers. The recommended values for the two bus control registers are contained in the ID ROM (refer to section on the ID ROM), but may require alteration for your particular application.

The HEPC2104 is shipped as standard with zero wait state SRAM. This is configured as 256 Kbytes, 512 Kbytes, 1 Mbyte, or 2 Mbytes on each of the Local and Global busses. Although it is possible to populate only one bank, we do not recommend it as this will prevent the use of any concurrent Local and Global bus accesses.

The HEPC2104 is also available with FLASH memory fitted to the Global bus in place of any SRAM. The Global FLASH is configured as 512 Kbytes, 1 Mbyte, or 2 Mbytes. When fitted with FLASH memory the HEPC2104 can be made to self boot from an application stored in Global memory.

The Local bus memory is addressed on Local Strobe 0 only, but also requires the TCLK0 pin to be driven low ('C44 Timer 0 Control Register: bit 0 set low, bit 1 set high, bit 2 set low) to be accessed. When the TCLK0 pin is driven high ('C44 Timer 0 Control Register: bit 0 set low, bit 1 set high, bit 2 set high) the Local bus is used to access the ID ROM. The standard ID ROM value for the Local Bus Control Register (LBCR) will cause the local SRAM to be addressed over and over again throughout the Local address range. This can be altered by setting the LBCR to other values.

The Global bus memory, whether fitted with SRAM or FLASH is addressed in hardware throughout the Global bus address range on STRB0 only, echoing as it goes. However it is possible to restrict this range by setting the correct registers in the 'C44 (refer to the 'C4x User Guide).

All SRAM will perform accesses as fast as the 'C44 can. This gives memory reads in 1 cycle and memory writes in 2 cycles.

The Global FLASH Memory Option

With FLASH memory fitted on the Global bus the HEPC2104 can be used to form a self-booting system, where the processor boots from an application stored in the Global memory. First a valid boot-sequence containing the self-booting program code must be written to the Global memory. With the Global FLASH programmed the 'boot' jumper can be fitted and the HEPC2104 reset. Following the system reset the processor will boot from the program previously stored in the FLASH memory. The following section describes the required operations when using the FLASH booting feature of the HEPC2104.

FLASH Booting

In order to correctly use FLASH memory fitted to the Global bus, the Global Bus Control Register (GBCR) must be set up to use the hardware ready signal. The hardware ready signal will guarantee correct operation of the FLASH write and FLASH read cycles. (Refer to the 'C4x User Guide for a description of the Bus Control Registers.) The value for the Global Bus Control Register of 1ef78000 (Hex) supplied in the IDROM will provide correct operation of the external Global memory hardware.

Note, the IDROM does not declare any Global memory when the board is fitted with Global FLASH. This is to prevent software tools such as 3L Parallel C from using the FLASH memory as general-purpose memory.

The first operation that must be performed when FLASH booting is to program the Global FLASH with a boot sequence that contains the self-booting application. When programming a boot sequence into the FLASH, data must be written in blocks equal to the sector length. The FLASH is organised in sectors, where one sector is programmed at a time. For a HEPC2104 fitted with 512 Kbytes of Global FLASH the sector size is 128 words. For a HEPC2104 fitted with 1 or 2 Mbytes of Global FLASH the sector size is 256 words. All memory accesses to the FLASH are in words of 32-bits in size.

If a word within a sector is to be changed, the data for the entire sector must be loaded into the FLASH memory. Any word that is not loaded during the programming of its sector will be erased to read ffffffff (Hex). Once the words of a sector are loaded they are simultaneously programmed during the internal programming period. After the first word location of a sector is written to the Global FLASH the successive locations within that sector must be written with each access within 150us of the previous access. If an access is not detected within 150us of the last access, the load period will end and the internal programming period will start. The words may be loaded in any order, sequential loading is not required. Once an internal programming operation has been initiated the next sector cannot be loaded until a minimum of 10ms later.

To be able to boot from the Global FLASH, the correct boot sequence must be written to Global memory starting from address 80000000h. The first location must contain the value 32, to specify the memory width (refer to the 'C4x User Guide). The rest of the data stream from 80000001h (incrementing by 1 for each new boot word) should be programmed as per the requirements of the boot stream as defined in the 'C4x User Guide and the sector programming requirements detailed above.

An example program is provided at the end of this manual that reads a 3L Parallel C

application and loads it into FLASH memory. The example highlights the use of the 3L ROM booting kit for creating a kernel for use with the self-booting 3L application.

When the FLASH memory has been loaded with the correct boot sequence the HEPC2104 can be made to boot from the stored application by fitting a jumper to the 'BOOT' header and applying a system reset. With the 'BOOT' jumper fitted the next time the 'C44 boots it will begin booting from Global memory starting at address 80000000h.

By using the auto-reset feature (refer to the TIM-40 Reset section) the HEPC2104 can be set up to automatically assert and de-assert the system reset on power-up. When used with the 'BOOT' jumper fitted (and the FLASH memory programmed) this enables the HEPC2104 to self-boot on power-up from the application previously stored in the Global FLASH.

The ID ROM is addressed at all locations on the local bus if the TCLK0 pin is driven high (or not driven, as there is a pull up resistor on the TCLK0 pin). The ID ROM is a serial device that provides one bit per read cycle on bit 0 of Local data. The data is supplied least significant bit first. The address is reset to the start of the ROM when TCLK0 is taken low again.

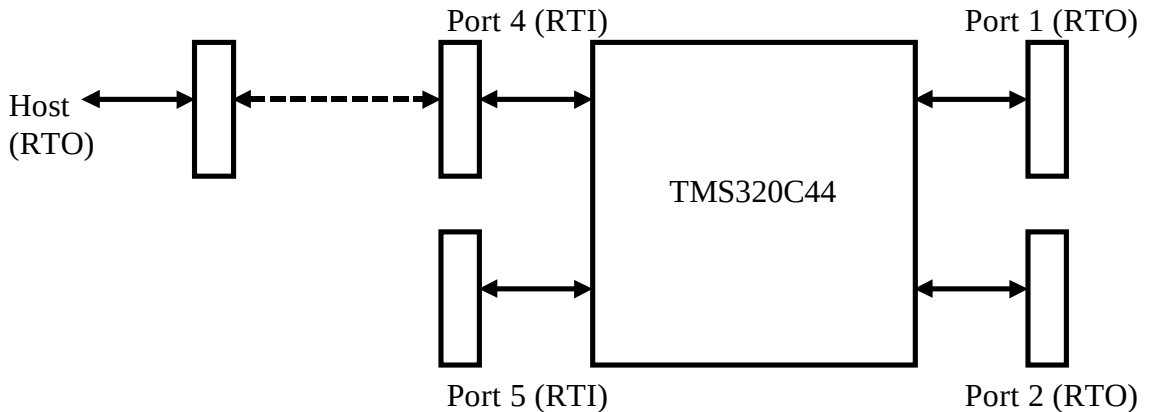
Please Note: Due to the IACK instruction that is executed by the bootstrap code it is possible for it to appear that the first bit from the serial ROM is missing. The only safe way to access the ID ROM is to first drive the TCLK0 pin low then high again before accessing the ID ROM.

The contents of the ID ROM is defined by the TIM-40 specification and is listed below.

word 0:	ID Size	0x13	- size excluding itself	
word 1:	Ram Address	0x2ff800		
word 2:	Table Size	0x13		
word 3:	Manufacturers ID	0x01	2 bytes	(01=Hunt Engineering)
	CPU ID	0x06	= 'C44	1 byte
	CPU Clock Rate	0x27	= 50MHz	1 byte
		or	0x20	= 60Mhz
word 4:	Module Function	0x0d	= HEPC2104	2 bytes
	Revision Level	0x0c	= Rev C	1 byte
	Reserved	0x0		1 byte
word 5:	GBASE0	0x80000000		
word 6:	GBASE1	0xffffffff		
word 7:	LBASE0	0x00300000		
word 8:	LBASE1	0xffffffff		
word 9:	GSIZE0	0x00010000	256 Kbytes (64Kwords)	
		or	0x00020000	512 Kbytes (128K words)
		or	0x00040000	1 Mbyte (256Kwords)
		or	0x00080000	2 Mbytes (512Kwords)
		or	0x0	no global memory
word 10:	GSIZE1	0x0		
word 11:	LSIZE0	0x00010000	256 Kbytes (64K words)	
		or	0x00020000	512 Kbytes (128K words)
		or	0x00040000	1 Mbyte (256K words)
		or	0x00080000	2 Mbytes (512K words)
		or	0x0	no local memory
word 12:	LSIZE1	0x0		
word 13:	FSIZE	0x800		(i.e. only on chip.)
word 14:	WAIT_G0	0x1	4 bits	
	WAIT_G1	0x0	4 bits	
	WAIT_L0	0x1	4 bits	
	WAIT_L1	0x0	4 bits	
	PWAIT_G0	0x1	4 bits	
	PWAIT_G1	0x0	4 bits	
	PWAIT_L0	0x1	4 bits	
	PWAIT_L1	0x0	4 bits	
word 15:	TIMER0_PERIOD	0x30d4	for 50MHz	(value to be loaded)

	or	0x3a98 for 60MHz	for 1ms tick)
word 16:	TIMER1_PERIOD	0x3c	
word 17:	TIMER 0 CNTRL	0x2c2	16 bits
	TIMER 1 CNTRL	0x2c1	16 bits
word 18:	GBCR	0x1ef78000	
word 19:	LBCR	0x1ef78000	
word 20:	AINIT SIZE	0x0	
word 21:	B SIZE	0x0	

In addition to the 'Host' Comport connection, the HEPC2104 also has four Comport connections to the 'C44. All of the Comports are buffered using the 2nd Generation HUNT ENGINEERING buffering scheme and are brought out to connectors.



The 'C44 Comport is a parallel communications bus which can transmit in either direction, but to change that direction a token must be passed between the communicating processors. This scheme relies on the fact that 2 of the 'C44s four Comports power up (or get reset) with a token and the other 2 power up (or get reset) without a token. Care must be taken to only connect a reset_to_input (RTI) Comport to a reset_to_output (RTO) Comport.

The cables available from HUNT ENGINEERING are polarised in such a way as to prevent incorrect connection. Also, 2nd generation Comport cables are polarised in a different way to previous Comport schemes, again preventing connection of incompatible Comport buffering schemes. Care should be taken to ensure that any cables provided by the user are similarly polarised.

The diagram above shows a dotted line between the host Comport and Port 4 of the 'C44. This is the 'Default' Comport connection. When connecting the host interface to the 'C44, the host Comport can be directly connected to Port 4 of the 'C44 by fitting a jumper to 'DEFLT' header. In this configuration a Comport cable must not be fitted between the 'Host' header and the 'Port 4' header.

If you connect both the Comport 'DEFLT' jumper and a Comport cable to the 'Host' and 'Port 4' Comport headers the Comport will become inactive and the 'DEFLT' LED will illuminate showing an error has been made in the set-up.

The IIOF0, IIOF1 and Timer 1 signals from the 'C44 are used in the following way.

1. IIOF0: The IIOF0 pin of the 'C44 is pulled high, buffered and brought to the 'IIOF0' LED. Whenever the TIM module drives the IIOF0 pin as a logic low output the 'IIOF0' LED will illuminate. Another buffered version of the IIOF0 signal is provided on pin 1 of the 'IIOFs' header.
2. IIOF1: Pin 3 of the 'IIOFs' header is buffered and driven on to the IIOF1 pin of the 'C44. As such, pin 3 is intended for use as a general purpose input or as an interrupt. Note, the drive on the IIOF1 pin is enabled following a rising edge on the IACK pin of the 'C44. When the IIOF1 buffer is not enabled the IIOF1 pin of the 'C44 is pulled high to guarantee it will be high at the time of booting. This is required to ensure the 'C44 will correctly boot from a Comport. To enable the buffer an IACK instruction to an address in external memory is required.

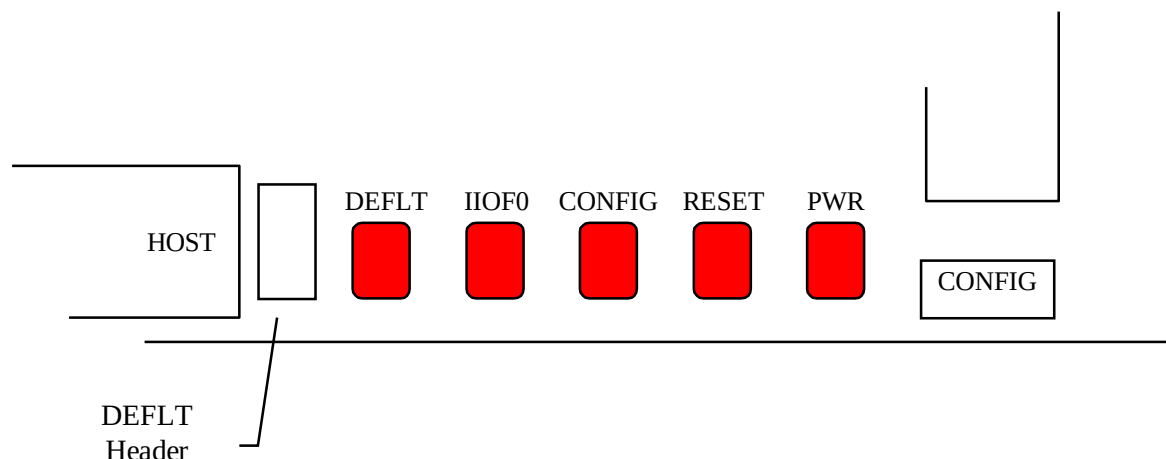
Note: the normal Comport boot sequence will perform an IACK instruction at the end of the sequence, causing a rising edge on the IACK signal as long as that IACK instruction uses an external memory address.

Once enabled the IIOF1 pin can be used as an general purpose input or as an interrupt (refer to the 'C4x User Guide).

Timer 1: The Timer 1 pin of the 'C44 is pulled high, buffered and driven out on pin 1 of the 'TCK' header. This signal is provided for I/O modules that may require a software programmable clock (refer to the 'C4x User Guide section on the Timer Registers).

This timer is intended to allow an SRAM module (DRAM modules use Timer1 for DRAM refresh) to provide a software controllable clock, which can be useful in many systems. For instance an A/D module such as the HEGD1 or HEGD2 can have a software controllable sample clock driven by this signal.

There are five LEDs on the HEPC2104. These provide an indication of the state of the HEPC2104 and other connected signals including the system wide TIM-40 Reset and Config signals. They are located in a row between the 'Host' Comport header and the Config header.



The first LED is the 'PWR' LED which will illuminate whenever the power-down logic is switched on, indicating that the 'C44 and associated logic has power on.

The second LED is the 'RESET' LED. This will illuminate when the boards reset is asserted (logic low). This reset is buffered from the reset selected on the centre pin of the 'RES SEL' header.

The third LED is the 'CONFIG' LED which will illuminate when the Config signal is asserted (logic low).

The fourth LED is the 'IIOF0' LED. It is a buffered version of the IIOF0 pin (refer to the previous section on I/Os and Timers) and is illuminated when IIOF0 is driven low.

The fifth LED is the 'DEFLT' LED which will illuminate when there is a selection error for the host Comport connection (refer to the section on the Host Comport Interface).

This section details the HEPC2104 hardware, giving connector pin-outs and signal level information. It is placed here as the 'system' configurer would not normally require information. The following sections are however necessary for a user wishing to develop compatible hardware, or for use in system fault finding.

General Information

The HEPC2104 is a standard PC/104 format board intended for applications where a system of TIM40s need to interface to a PC/104 host machine. The HEPC2104 conforms to the PC/104 Specification, version 2.1, for 16-bit modules.

All Comports on the HEPC2104 are buffered using the HUNT ENGINEERING 2nd generation Comport buffering scheme. As such, the Comports must not be used to connect to earlier Comport schemes, or Comport schemes from another vendor.

In particular HUNT ENGINEERING have carried out many experiments and believe that the Comport connections **MUST** be buffered in all circumstances. This process of course reduces the performance in terms of speed, but provides essential reliability.

Comport Connectors

To allow interconnection between boards it has been found that the Comports must be buffered and that care must be taken in cabling these Comport connections. A polarised scheme for providing a buffered Comport at a connector is shown below.

		*			
*CSTRB	1	O	O	2	Ground
*CRDY	3	O	O	4	Ground
D0	5	O	O	6	TERMGND 0
D1	7	O	O	8	TERMGND 1
D2	9	O	O	10	POLARISE 0
D3	11	O	O	12	Ground
D4	13	O	O	14	Ground
*CREQ2	15	O	O	16	Ground
D5	17	O	O	18	POLARISE 1
D6	19	O	O	20	Ground
D7	21	O	O	22	Ground
*CREQ1	23	O	O	24	Ground
*CACK	25	O	O	26	Ground

The polarisation of the on-board connectors is as follows.

POLARISE 0	POLARISE 1	
Cut Short	Pin	Reset_to_output Comport connector
Pin	Cut Short	Reset_to_input Comport connector

All Comport signals are parallel terminated using 220 Ohms to +5V and 330 Ohms to Ground. This termination network dissipates significant power and the resistor packs used are often hot to the touch.

In order to limit the power consumption of these resistor packs when the Comport is not being used, the ground pin of the parallel terminators, and the enable pin of the data buffers has been placed on the connectors. Thus when there is no cable fitted the power consumption of that comport is reduced. However when a (one to one) cable is fitted between an RTI connector and an RTO connector, each TERMGND signal is grounded by the Ground connection at the other end of the cable.

The TERMGND pins are allocated as follows.

TERMGND 0	TERMGND 1	
buffer and Rpack enable	Gnd	Reset_to_output Comport connector
Gnd	Buffer and Rpack enable	Reset_to_input Comport connector

The CREQ1 and CREQ2 signals are the only signals that are not bi-directional but are a buffered permanent representation of the CREQ signal at the 'C4x pin at that end of the comport connection. This enables the buffering scheme to delay the direction switch of the buffers until the signals have been driven to the far 'C4x, regardless of cable length.

The CREQ pins are allocated as follows.

CREQ1	CREQ2	
Driven	Input	Reset_to_output Comport connector
Input	Driven	Reset_to_input Comport connector

The buffering scheme used on the HEPC2104 is the 2nd generation HUNT ENGINEERING Comport buffering scheme. This has the advantage of increased bandwidth and reduced power consumption. This scheme is NOT compatible with the first generation scheme.

It should be noted if trying to connect a non HUNT ENGINEERING product to this connector, that the signals are not inverted from the 'C4x signals but that they are terminated in an equivalent 100 Ohms and that for correct operation the product being connected should both provide this termination and have sufficient drive capability to drive this termination on the HEPC2104. If you are attempting to design your own hardware to interface with the HEPC2104 please ask for the application note that covers this subject.

The signals are all driven using FCT245AT or FCT244AT buffers with 64mA drive capability, and are essentially TTL levels but with the effective 100 Ohms termination at both ends of the cable.

The 'RES SEL' Header

The 'C44 on the HEPC2104 is reset by the reset selected onto the centre pin of the 'RES SEL' header. This reset must be taken low to assert the reset and taken high again to de-assert the reset. This signal must be driven at all times as a floating reset signal will cause spurious operation.

This board reset is buffered and driven out on the 'Reset Output' pin (pin 3) of the 'CONTROL OUT' connector. This buffered output can be used to drive further slave boards without fan-out problems. This output is a TTL level signal suitable for connecting directly to the reset input of any HUNT ENGINEERING boards.

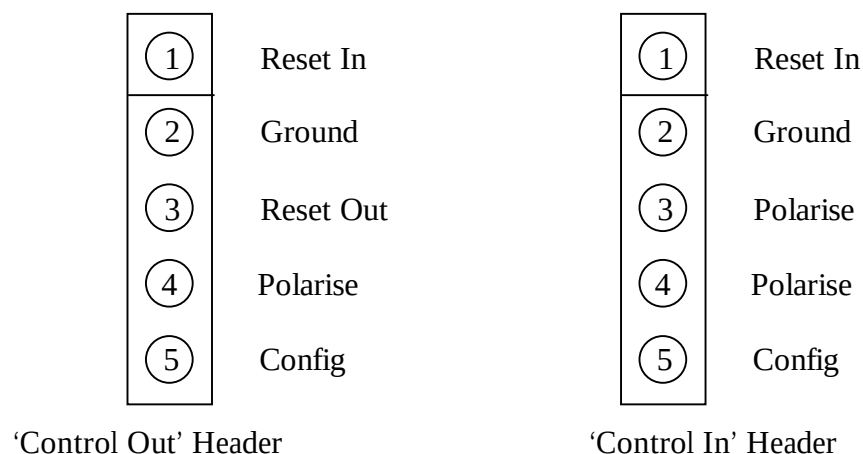
The board reset can be selected from three sources (refer to the section on TIM-40 Reset).

1. Host Reset: The host reset is buffered from bit 0 of the Control Register interfaced to the ISA bus. The host reset is the same sense as the register bit, and is asserted when driven low (bit 0 written as zero) and de-asserted when driven high (bit 0 written as one).
2. Reset In: The reset in signal is a TTL buffered input on the 'CONTROL IN' connector. If left disconnected the reset will be pulled high (inactive). This input is suitable for connection to the reset output of any HUNT ENGINEERING boards.
3. Auto Reset: The auto reset is a power-up reset that becomes asserted for a short time following the power-up of the power-down logic on the HEPC2104. On power up the auto reset is asserted low for approximately 64msecs, and is then de-asserted high.

Control Connectors

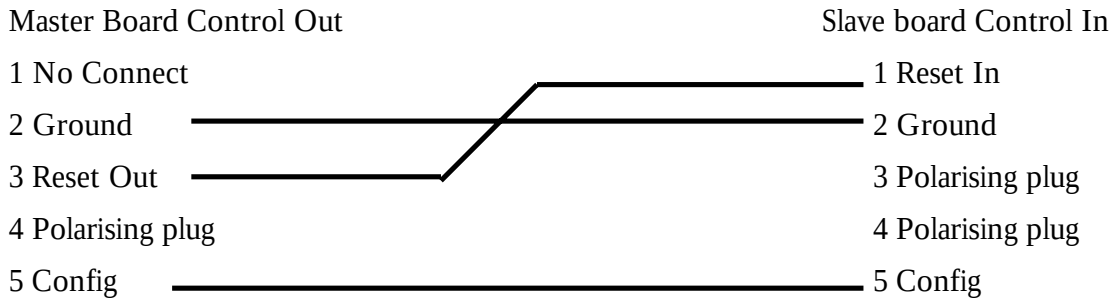
In a system of connected 'C4x processors there must be a single reset used in order to maintain the correct directions of the Comports. There should also be a system wide connection of the open collector Config signal.

The Control connectors provided on the HEPC2104 are pinned out in a manner that is standard to all HUNT ENGINEERING motherboard products.



Interconnecting Motherboards

Standard HUNT ENGINEERING control cables can be used to connect multiple boards together as follows.



Further slave boards can be added in the same way, chaining 'CONTROL OUT' connectors to 'CONTROL IN' connectors using standard cables as above, as long as the Reset and Config jumpers are configured correctly.

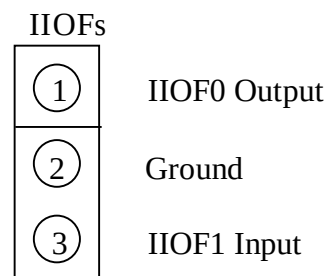
If the HEPC2104 is to be connected as a slave to another HUNT ENGINEERING motherboard the 'CONTROL IN' connector can be connected to the 'CONTROL OUT' connector of another motherboard, using a HUNT ENGINEERING standard control cable.

If there are further HEPC2104 modules connected in the same system as these ones, then the 'CONTROL OUT' header is provided to enable a HUNT ENGINEERING standard control cable to be used to connect the control of this HEPC2104 to the 'CONTROL IN' of the next board in the system.

The Control headers will accept many connector types. An example of a suitable connector is the DuPont 65039-32 housing with DuPont 47712-002 crimps. Note, the HEPC2104 has a silk-screen box around the header to indicate the location of pin 1, in addition to the polarising pins.

The IIOFs Header

The 'IIOFs' header provides the IIOF0 pin of the 'C44 as a buffered output and the IIOF1 pin as a buffered general purpose input or interrupt.



The 'IIOFs' header will accept many connector types. An example of a suitable connector is the DuPont 65039-034 housing with DuPont 47712-002 crimps. Note the header has a box around pin 1.

The buffered IIOF0 signal on pin 1 is a TTL output and can drive:

$I_{ol} = 48\text{mA}$ (Vout = 0.5V max)

$I_{oh} = -48\text{mA}$ (Vout = 2.0V min)

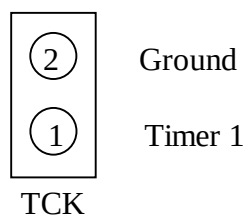
The HIOF1 Input signal on pin 3 is buffered with a TTL gate that requires:

$I_{il} = -0.5\text{mA}$ (Vin = 0.5V max)

$I_{ih} = 0.02\mu\text{A}$ (Vin = 2.0V min)

The Timer 1 Header ('TCK')

The 'TCK' header provides a buffered 'C44 Timer 1 output on pin 1.



Note, pin 1 of the 'TCK' header is located on the side nearer the PC/104 connectors. The 'TCK' header will accept many connector types. An example of a suitable connector is the DuPont 65039-035 housing with DuPont 47712-002 crimps.

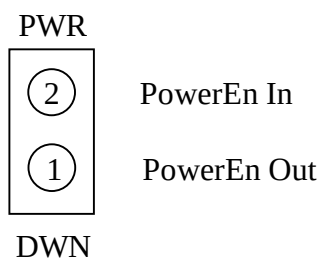
The Timer 1 output is a TTL output and can drive:

$I_{ol} = 48\text{mA}$ (Vout = 0.5V max)

$I_{oh} = -48\text{mA}$ (Vout = 2.0V min)

The 'PWR DWN' Header

The 'PWR DWN' header enables the user to power down and power up under software control the majority of the logic on the HEPC2104 (refer to the section on the Power Down Feature). In a single board system, this simply requires fitting a jumper between pins 1 and 2 of the 'PWR DWN' header. In a multiple HEPC2104 system, a cable is required that connects pin 1 of the first controlling 'PWR DWN' header to both pin 2 of that first header, and pin 2 only of all other 'PWR DWN' headers.



Note, pin 1 of the 'PWR DWN' header is located on the side nearer the PC/104 connectors. The 'PWR DWN' header will accept many connector types. An example of a suitable connector is the DuPont 65039-035 housing with DuPont 47712-002 crimps.

The PowerEn Output signal on pin 1 is a buffered TTL output that can drive:

$I_{ol} = 50\text{mA}$ (Vout = 0.5V max)

$$I_{oh} = 50\text{mA} \quad (V_{out} = 2.0\text{V min})$$

The PowerEn Input signal on pin 2 is pulled up and connected to the gate of an FET that requires:

$$I_{il} = 5\mu\text{A} \quad (V_{in} = 0.5\text{V max})$$

$$I_{ih} = 5\mu\text{A} \quad (V_{in} = 2.0\text{V min})$$

JTAG connector

This 'JTAG' slave interface is used to debug (including break-pointing etc.) and to reset individual processors or even individual parts of processors. A JTAG master such as a HUNT ENGINEERING HEPC2E, HESB40 or an XDS-510 with appropriate cabling is required.

The 14-pin 'JTAG' connector will accept a standard 14-way cable from an XDS-510 or compatible debugger. The pin-out of the connector is shown below.

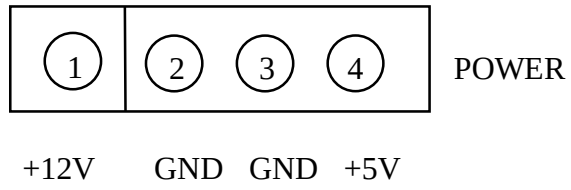
		*			
TMS	1	O	O	2	TRST
TDI	3	O	O	4	Ground
Presence Detect	5	O	X	6	Polarisation
TDO_RETURN	7	O	O	8	Ground
TCK_RETURN	9	O	O	10	JTAG TERM Ground
TCK	11	O	O	12	Ground
EMU0	13	O	O	14	EMU1

The cable length of the JTAG connection should be kept as short as possible to prevent noise or cable delay becoming a problem. HUNT ENGINEERING have found that lengths of up to 1 metre are possible. The connector pin-out can also be found in Appendix B of the 'C4x User Guide.

Power Connector (Optional)

There is an optional site for a power connector on the HEPC2104, allowing it to be powered in 'stand alone' mode. It is not normally fitted to the HEPC2104, but can be fitted by special request. If fitted will accept a standard 3.5" floppy drive power connector from a PC power supply, or can be cabled as follows using a connector such as Molex 0.1" pitch KK connector, part number 22-01-2045 with crimp inserts 08-050-0032, which accepts wire between 22 and 30 AWG

The pinout is as follows:



Physical Dimensions of the HEPC2104

A PC/104 card is 3.775 inches by 3.55 inches with an extra envelope for the Connectors. The HEPC2104 fully complies to the PC/104 standard for 16-bit modules.

Power Requirements of the HEPC2104

The HEPC2104 only uses power from the 5V supply, the amount will vary slightly with the amount of memory accesses taking place.

The maximum rating of the HEPC2104 is 2.8A, but the typical (and measured) is nearer 0.9A with all circuitry power up. An extra 0.1A will be drawn by each end of each Comport connector that is fitted.(but no more for the use of the 'DEFLT' jumper for the host comport connection.)

With the main circuitry powered off, the maximum is 0.25A but the typical (and measured) is nearer 0.15A.

In most cases development tools will have been supplied for the relevant Host machine Operating System. These tools will either use the Application Programming Interface (API) provided in the libraries and/or drivers supplied by HUNT ENGINEERING, or will directly access the HEPC2104.

It is important to note that the HEPC2104 is totally software compatible with the first Comport interface of the HEPC2E, and all tools for the HEPC2E Port A interface will function on the HEPC2104. In fact the HUNT ENGINEERING API should be set to use the HEPC2E, whenever you use the HEPC2104. The only exception to this is the programming of the Power Down Enable in the Control register which is not featured on the HEPC2E.

The following sections describe briefly the sort of tools supported by the HEPC2104, for more details please consult the PC development tools and API user manual.

Development Tools

There are three main development tools that are supported by HUNT ENGINEERING for the HEPC2104 which are described in the following sections.

3L Parallel C

The Parallel C compiler from 3L is based upon the TI compiler for the 'C4x. It provides a means to load multiple communicating sequential processes onto single or multi-processor systems. A kernel is loaded by the system onto each processor enabling it to provide kernel services such as access to the resources of the host machine via STDIO libraries, task scheduling and inter-process communication via a Virtual Channel Router.

HUNT ENGINEERING Server/Loader

The HUNT ENGINEERING Server/Loader provides a means to load single sequential programs onto each processor in a system. The code can be generated using either the TI C compiler or assembler.

Access to the Host machine resources via a STDIO library is provided to the 'C4x that is connected directly to the host via a Comport.

Go DSP Code Composer

The Code composer product from Go DSP is an integrated development and debug environment for multiple 'C4x systems.

As it requires a board with a JTAG Master interface, it cannot be used directly with the HEPC2104. In order to use the Go DSP tools, the HEPC2104 will need to be used with another board such as the HEPC2E or HEPC4. The Go DSP tools can also use either the TI C compiler/assembler or 3L Parallel C with the host Comport interface in addition to being used via the JTAG debugging chain.

Application Programming Interface (API)

The API that is provided by HUNT ENGINEERING for the HEPC2104 enables a program on the Host computer (perhaps one of the development tools listed above) to communicate with the HEPC2104 (or other API supported HUNT ENGINEERING motherboards) via a consistent and easy to use software interface. The same interface is provided on many different platforms in order to make the program using it portable to different Host machines and Operating systems.

It provides an 'open', 'close', 'read', 'write' and 'reset' interface with some additional features to properly support different I/O models and windows environments. The API interface is constant for different PC supported HUNT ENGINEERING products, and supports several operating systems such as MS Windows (3.11, 95 and NT).

In most cases the API will require the use of an interrupt, in order to make efficient use of the operating systems resources. The installation procedure will prompt you to choose the settings of the HEPC2104 host interface, and these choices must be reflected in the hardware configuration settings.

Software Requirements of the HEPC2104

There are a few requirements that the hardware places on the software to enable the HEPC2104 to function correctly. These are detailed in the following sections.

The Host Interface

A feature of the 'C4x Comports can cause an apparent lock up to occur. The situation that can cause this is as follows.

1. Firstly, with the connected 'C4x Comport not ready to empty its input FIFO, and with the PC/104 host starting to send data to the Comport, the 'C4x input FIFO will become full.
2. Next, at the same time the 'C4x is sending a large amount of data to the host Comport interface, and as a result the 'C4x end of the Comport will not relinquish its token to the PC/104 host.

If the buffer space on the HEPC2104 is allowed to fill up in both directions, then the PC/104 transfer cannot complete because the buffer is full, and the 'C4x Comport transfer is locked up because the buffer between the 'C4x and the PC/104 is full. No further transfer can take place.

The only way to recover from this situation is for the PC/104 to read its' buffer. This requires the PC/104 host machine to multi-task. The PC/104 software must take care of this in its' own manner. An interrupt feature is provided (refer to the section on Interrupts) which can help in preventing the problem.

The above problem is exactly the same as that which would occur between two communicating 'C4x processors, except that eventually one of the processors would read its' input FIFO. The HEPC2104 aims to prevent this deadlocking situation by requesting the token from the 'C44 when the buffer nears full. This is entirely effective as long as the processors' input FIFO is not full, and will thus release the token.

Configuring for use with 3L Parallel C

The 3L Parallel C compiler loads a kernel onto the C40 at boot time. This kernel (TIM-40.krn) essentially loads the ID ROM loader into internal memory, configures the C40 as per the ID ROM and then continues to boot the application. Thus the user does not need to worry about the initialisation as it is done for them.

Parallel C is based on the TI C compiler and hence can use any code written for the TI compiler but provides extensions to the compiler.

The user can govern which parts of the TIM-40 memory the various pieces of code and data get placed, using an OPT statement. This statement is fully described in the 3L manual, but its basic function is to allow you to place any of {CODE (program), STATIC (global and external variables), STACK (variables declared within a function), HEAP (memory available to functions such as malloc), DATA (combined stack and heap)} into one of {RAMBLK0 (internal ram block 0), RAMBLK1 (internal ram block 1), LOCAL0 (the memory on Local bus strobe 0), LOCAL1 (the memory on local bus strobe 1. N.B. none fitted on this module), GLOBAL0 (the memory on Global bus strobe 0), GLOBAL1 (the memory on the Global bus strobe 1. N.B. none fitted on this module)}.

3L provide a “Debugger Support Kit” which allows you to interface your own application code with the C source debugger. When using this system, you do not need the initialisation to be done by your ‘.cmd’ command file for the debugger. You also do not want to have a ‘reset’ in your ‘.cmd’ file.

As the HEPC2104 is compatible with the HEPC2E the environment variable TISLINK should be set as follows:-

```
Set TISLINK=HEPC2E, IO:200
```

The IO option informs the 3L server TIS that the base address of the HEPC2104 is at 200hex, in this example. The valid base addresses to pass are 160,200 or 300 (hex). Note, if the IO option is left off, the TIS server defaults to address 200 hex.

e.g. `set TISLINK=HEPCE`

is identical to

```
set TISLINK=HEPC2E, IO:200
```

and the kernel variable should be set as follows :-

```
set TISR00T=TIM40.KRN
```

adding the path to this file if necessary.

This can be done by using the SET command at the DOS prompt, or more conveniently by adding the SET command lines into the ‘autoexec.bat’ file.

For other addressing options refer to the 3L technical note for the HEPC2E.

Configuring for use with the Server/Loader

To configure the host system to use the Server/Loader the network description file used by the Server/Loader can be set up in one of two ways as detailed below.

1. Direct I/O (NOT RECOMMENDED): In the network description file the board type field must be set to HEPC2EA, and the address field must be set to 160, 200 or 300 according to the setting of the base address jumpers.

2. API (RECOMMENDED): In the network description file the type file must be set to API, the board_type field must be set to hep2e and the device_id field must be set to 0.

Using the JTAG Debugger with other Tools

If you are using the JTAG Debugger along with other Comport access programs such as 3L Parallel C or the 'C4x Loader/Server you will need to ensure that the processor is free running after finishing a session using the debugger.

This can be done by either typing the command 'runf' (run free) within the debugger before quitting, or by typing 'emurst' at the DOS prompt after quitting. If this is not done, the debugger maintains control of the processor which will therefore not register an active reset when you next try to boot it.

Using the TI Compiler/Assembler

If you are using the TI compiler or assembler, you must remember to set the IIOF pins, TCLK0 pin and bus control registers explicitly. This can be done by addressing the registers directly from the code and initialising them to the correct values, or it can be done by reading the values from the ID ROM and using them. The second option of reading the ID ROM would also allow the code to be run on another TIM-40 with different requirements without having to change the code.

Your code uses memory as defined in a '.cmd' file which allows you to define your memory areas and assign the sections{.text (code), .cinit (initialisation tables), .const (constants), .stack (stack), .sysmem (dynamic memory or heap), .bss (variables)} to those areas you have defined.

It must be remembered that until the register initialisation has occurred, any external memory cannot be used because Local bus addresses are taken as ID ROM accesses, and the bus control register for the Global bus might be incorrect. Hence it is usual to boot the initialisation code into internal memory and execute it before loading the main code. (The 'C4x Server/Loader software from HUNT ENGINEERING does this for you.)

An example of such an assembly code routine is given below.

```
; TITLE 'Assembly example'
;
; .global RESET, INIT
;
RESET .sect "reset_adr"
      .word INIT

      .data
STATUS .word 00001800H
TIMER0CTRL .word 0
TIMER0PER .word 0
TIMER1CTRL .word 0
TIMER1PER .word 0
GLOBALCTRL .word 0
LOCALCTRL .word 0
GBASE0 .word 0
GBASE1 .word 0
LBASE0 .word 0
LBASE1 .word 0
```

```

GSize0      .word      0
GSize1      .word      0
LSize0      .word      0
LSize1      .word      0

```

```

        .text

```

* This section is very important in setting up the Data Page
 * Pointer, Status Register, IVTP, TVTP, Internal Interrupt
 * Enable Register and Stack Pointer.

```

INITLDP      002F0000H,DP      ;16lsbs of DATA PAGE POINTER ;form
                                ;16msbs of inst address.(addr ;002F)
        LDI      @STATUS,ST      ;clear status register and clear
                                ;and enable the internal cache

        LDHI      002FH,AR0
        OR        0FC00H,AR0
        LDPE      AR0,IVTP      ;set IVTP to 002FFC00H
        LDHI      002FH,AR0
        OR        0FE00H,AR0
        LDPE      AR0,TVTP      ;set TVTP to 002FFE00H
        LDI      0,IIE          ;disable all internal interrupts
        LDHI      002FH,SP      ;Setting the Stack Pointer must
        OR        0FF00H,SP      ;be done in your program
        LDHI      0010H,AR0      ;set AR0 to 00100000H, this is
                                ;the address for INTERNAL
                                ;PERIPHERALS
        LDHI      0030H,AR1      ;set AR1 to 00300000H, this is
                                ;the address for the LOCAL BUS
        LDHI      08000H,AR2      ;set AR2 to 80000000H, this is
                                ;the address for the GLOBAL BUS

```

* This next section sets up the Timer0 Control Register and
 * Global and Local Memory Interface Control Registers. Although
 * the data in the ID ROM will be used to set up all three of
 * these registers, they need to be set here or you will not be
 * able to read the ID ROM anyway.

```

        LDHI      0C000H,R0      ;toggle tclk0 to reset ID ROM
        OR        02C2H,R0      ;this is done by writing
        STI      R0,*+AR0(20H)  ;C00002C2H to the Timer0 Control
                                ;Register,
                                ;pointed to by value in AR0 + 20H
        NOP
        NOP
        NOP
        LDHI      0C000H,R0      ;rewrite Timer0 Control Register
                                ;with C00002C6H
        OR        02C6H,R0
        STI      R0,*+AR0(20H)
        LDHI      1E4AH,R0      ;also, a valid setting must be
        OR        4000H,R0      ;loaded into the Global and Local
        STI      R0,*+AR0(0)    ;Memory Interface Control ;Registers
        STI      R0,*+AR0(4)    ;now we can start reading ID ROM

```

* This section sets up the 'C4x according to the ID ROM data.
 * The IMPORTANT REGISTERS TO SET ARE:
 * Timer0Period, Timer1Period, Timer0Control, Timer1Control, and
 * Global and Local Memory Interface Control Registers and the
 * IIF register Config bit.
 * The rest (i.e. Local/Global Size and Addresses) will be

* useful for storing address locations for your application to
 * use

```

CALL    IDWORD          ;word 0      (see subroutine IDWORD)
CALL    IDWORD          ;word 1
CALL    IDWORD          ;word 2
CALL    IDWORD          ;word 3
CALL    IDWORD          ;word 4      (not interested in words
CALL    IDWORD          ;word 5      0 to 4)
STI     R2,@GBASE0      ;store GBASE0 value from ID ROM
CALL    IDWORD          ;word 6
STI     R2,@GBASE1      ;store GBASE1 value from ID ROM
CALL    IDWORD          ;word 7
STI     R2,@LBASE0      ;store LBASE0 value from ID ROM
CALL    IDWORD          ;word 8
STI     R2,@LBASE1      ;store LBASE1 value from ID ROM
CALL    IDWORD          ;word 9
STI     R2,@GSIZE0      ;store GSIZE0 value from ID ROM
CALL    IDWORD          ;word 10
STI     R2,@GSIZE1      ;store GSIZE1 value from ID ROM
CALL    IDWORD          ;word 11
STI     R2,@LSIZE0      ;store LSIZE0 value from ID ROM
CALL    IDWORD          ;word 12
STI     R2,@LSIZE1      ;store LSIZE1 value from ID ROM
CALL    IDWORD          ;word 13
CALL    IDWORD          ;word 14
CALL    IDWORD          ;word 15
STI     R2,@TIMER0PER    ;store TIMER0PER value from ID ROM
STI     R2,*+AR0(28H)    ;timer 0 period register
CALL    IDWORD          ;word 16
STI     R2,@TIMER1PER    ;store TIMER1PER value from ID ROM
STI     R2,*+AR0(38H)    ;timer 1 period register
CALL    IDWORD          ;word 17
LDI     R2,R1
AND     0FFFFH,R2        ;retain      bottom      16      bits
                                ;(timer0ctrl)
LDHI    0C000H,R3        ;top two bits must be set to allow
                                ;timer0 to continue running when
OR       R3,R2            ;you are debugging your programs.
STI     R2,@TIMER0CTRL    ;store TIMER0CTRL from ID ROM
LSH     -16,R1            ;TIMER0CTRL cannot be used yet as
LDHI    0C000H,R3        ;we still need to read ID ROM
OR       R3,R1
STI     R1,@TIMER1CTRL    ;store TIMER1CTRL from ID ROM
STI     R1,*+AR0(30H)    ;timer 1 control register
CALL    IDWORD          ;word 18
STI     R2,@GLOBALCTRL    ;store GLOBALCTRL from ID ROM
STI     R2,*+AR0(0)      ;glob mem interface control reg
CALL    IDWORD          ;word 19
STI     R2,@LOCALCTRL    ;store LOCALCTRL from ID ROM
STI     R2,*+AR0(4)      ;local mem interface control reg
LDI     @TIMER0CTRL,R0
STI     R0,*+AR0(20H)    ;Timer 0 Control Register can be
                                ;set now as we have finished
                                ;reading ID ROM (for now!)
LDI     IIF,R0
AND     2444H,R0
OR      2444H,R0        ;Store 2444H to IIF, to set the
                                ;config bit.
                                ;This bit (bit 3) will then tell

```

```

PC      LDI      R0,IIF          ;that the 'C40 has booted

* ===== End of initialisation =====

      LDHI      0C000H,R0        ;Reset ID ROM again to go back to
      ADDI      02C2H,R0        ;the beginning of the ID ROM
      STI      R0,*+AR0(20H)
      NOP
      NOP
      NOP
      OR      02C6H,R0
      STI      R0,*+AR0(20H)

* ===== Get a word from ID ROM =====

IDWORD: LDI      0000H,R2        ;read 32 serial bits to form a word
      LDI      31,RC            ;use the repeat counter
      RPTB      BIT
      LSH      -1,R2
      LDI      *AR1++,R1
      LSH      31,R1

BIT: ADDI      R1,R2
      RETSU

      .end

```

The C Source Debugger

If you are using the C source debugger to debug your code in either source or assembler, you can load the code onto the 'C4x in two ways.

If you are loading the code using the debugger then you can set all of the registers up using the command file.

A command file for this module would normally have the following entries.

```

mr
ma      0x00000000,0x800 ,rom    ;2K EPROM LOCAL, STRB0
ma      0x002ff800,0x400 ,ram    ;INTERNAL BLK 0
ma      0x002ffc00,0x400 ,ram    ;INTERNAL BLK 1
ma      0x00300000,0x80000,ram   ;2Mb of SRAM LOCAL,
ma      0x00100000,1,ram         ;local port control
ma      0x00100004,1,ram         ;global port control
ma      0x00100020,9, ram        ;Timer 0
ma      0x00100030,9, ram        ;Timer 1
ma      0x001000A0,9, ram        ;DMA 0
ma      0x001000B0,9, ram        ;DMA 1
ma      0x001000C0,9, ram        ;DMA 2
ma      0x001000D0,9, ram        ;DMA 3
ma      0x001000E0,9, ram        ;DMA 4
ma      0x001000F0,9, ram        ;DMA 5
;don't map in the comports as the display of these registers
;effectively reads the data and causes it to be lost!

reset
e *0x100004=0x1EF78000          ;set local bus control reg for

```

```

                                ;SRAM
e *0x100000=0x1EF78000        ;set global bus control reg for
                                ;SRAM

e *0x100020=0xC00002C2        ;set timer0 control register,
                                ;setting top 2bits to one to
                                ;prevent the debugger halting the
                                ;timers. This drives the TCLK0 pin
                                ;low to enable the local SRAM.
e *0x100030=0xC00002C1        ;set timer1 control register,
                                ;setting top 2 bits to one to
                                ;prevent the debugger halting the
                                ;timers.

dasm pc

```

Do not forget to release the Config signal at the beginning of your code execution by writing to the IIF register and setting the IIOF3 pin as a logic low output.

If you are using the JTAG Debugger along with other tools such as 3L Parallel C or the HUNT ENGINEERING 'C4x Loader/Server to boot the code over a Comport, then you will not want the debugger to 'reset' the module or its registers, and so will not have entries in the command file ('.cmd' file) to access the set up registers.

When you have finished a debugging session but wish to continue to use 3L Parallel C or the HUNT ENGINEERING 'C4x Loader/Server you will need to ensure that the processor is free running after finishing a session using the debugger.

This can be done by either typing the command 'runf' (run free) within the debugger before quitting, or by typing 'emurst' at the DOS prompt after quitting. If this is not done, the debugger maintains control of the processor which will therefore not register an active reset when you next try to boot it.

Example Software for PC/104 Host Software Development

For most users, the development tools used will directly access the HEPC2104 removing you from the issues of communicating with the Host Interface of the HEPC2104. It is in fact STRONGLY recommended that any application developed for the HEPC2104 should use the HUNT ENGINEERING API, so that any future hardware changes or upgrades will not cause the application to be re-written, merely a new version of the API used.

However, for those users who need to develop an application that runs on the PC/104 machine to communicate with the HEPC2104 Host Interface, the following C program example is provided. This program illustrates the points that must be considered when accessing the HEPC2104 and transferring data to and from the Host Comport.

The program contains some useful functions, and a very basic *main()* function that sets up the I/O base address.

```

/*****
/*
/*      HEPC2104 Example Software      */
/*
/*      */
*****/
#include <string.h>
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
#include <dos.h>
#include <int.h>
#include <time.h>

int iobase;                                /* HEPC2104 Base Address */

/*****
/*
/*      MAIN()      */
/*
/*      */
*****/
int main()
{
void host_reset(void);
void power_down(void);
void power_up(void);
void writeport(unsigned int,unsigned long);
unsigned long readport(unsigned int);

/* Get the I/O Base Address */
iobase = -1;
while ((iobase != 0) && (iobase != 160) && (iobase != 200) &&
      (iobase != 300)){
    printf("Enter the I/O Port Base Address: 160,200 or 300. [200]");
    gets(string);
    iobase = atoi(string);
}
if ((iobase==0) || (iobase==200)) iobase = 0x200;    /* Default */
if (iobase==160) iobase = 0x160;
if (iobase==300) iobase = 0x300;
printf("I/O Base Address = %x\n",iobase);
/* Place required calls here to any of the functions listed below */

return EXIT_SUCCESS;
}

/*****
/*
/*      WRITE_DATA()      */
/*
/*      */
*****/

```

```

/*      This function write num_words 32-bit words to the Host      */
/*      Comport interface. It checks all of the FIFO flags to see   */
/*      how many words can be done. When FIFO space is available    */
/*      it transfers the correct number of words.                    */
/*                                                                    */
/*      The pointer *data points to the beginning of the data block */
/*      to be sent.                                                  */
/*                                                                    */
/*****
void write_data(unsigned long num_words,unsigned long *data)
{
int flags;
unsigned long i,n;
unsigned long ntotal = 0;

/* Do this loop until all words have been written */
while (num_words > 0){
/* Read the FIFO flags register */
flags = inpw(iobase+4);
/* Wait until PC/104 to Comport FIFO becomes not full */
while ((flags & 0x01) == 0) flags = inpw(iobase+4);

/* Check how many words can be written */
n = 0;
if ((flags & 0x40) == 0) n = 64;           /* fifo empty */
else if ((flags & 0x80) == 0) n = 60;      /* almost empty */
else if ((flags & 0x02) != 0) n = 4;       /* not almost full */
else if ((flags & 0x01) != 0) n = 1;      /* not full */

if (num_words < n) n = num_words;

for (i=0;i<n;i++){
    outpw(iobase,(data[ntotal + i] & 0xffff));
    outpw(iobase,((data[ntotal + i] >> 16) & 0xffff));
}
ntotal = ntotal + n;
num_words = (num_words - n);
}

return;
}

/*****
/*
/*      READ_DATA()
/*
/*      This function reads num_words 32-bit words from the Host
/*      Comport interface. It checks all of the FIFO flags to see
/*      how many words can be done. When words become available
/*      for reading, it transfers the correct number of words.
/*
/*      The pointer *data points to the beginning of the data block
/*      in which to place the read words.
/*
/*****
void read_data(unsigned long num_words,unsigned long *data)

```

```

{
unsigned long word;
int flags;
unsigned long i,n,ntotal;
ntotal = 0;
n = 0;

/* Do this loop until all words have been written */
while (num_words > 0){
    n = 64;
    /* Read the FIFO flags register */
    flags = inpw(iobase+4);
    /* Wait until Comport to PC/104 FIFO becomes not empty */
    while ((flags & 0x04) == 0) flags = inpw(iobase+4);

    if ((flags & 0x10) == 0) n = 64;          /* fifo full */
    else if((flags & 0x20) == 0) n = 60;     /* almost full */
    else if((flags & 0x08) != 0) n = 4;      /* not almost empty */
    else if((flags & 0x04) != 0) n = 1;     /* not empty */

    if (num_words < n) n = num_words;

    for (i=0;i<n;i++){
        word = (inpw(iobase) & 0xffff);
        result[ntotal + i] = (word |
                               ((inpw(iobase) >> 16) & 0xffff));

        ntotal = ntotal + n;
        num_words = (num_words - n);
    }
}

return;
}

/*****
/*
/*      HOST_RESET()
/*
/*      This function asserts the host reset signal on the HEPC2104.*/
/*      By connecting the 'Res Sel' jumper correctly this reset can */
/*      drive the system reset line.
/*
/*      Note: When writing to the Control Register, the other bits */
/*      in the register control the Endian FIFO setting and the */
/*      Power Down Enable will also be effected. This function */
/*      assumes that the Endian bit will be left set for little */
/*      endian operation, and the power down logic will be left in */
/*      powered up state.
/*
*****/
void host_reset(void)
{
    outpw(iobase+6,0x00);          /* Assert the host reset */
    outpw(iobase+6,0x01);          /* De-assert the host reset */

return;
}

```

```

}

/*****
/*
/*      POWER_DOWN()
/*
/*      To turn off the power-down logic on the HEPC2104, the power */
/*      enable bit (bit 2) of the Control register (offset 6), must */
/*      be set high.
/*
/*
*****/
void power_down(void)
{
    outpw(iobase+6,0x04);          /* Set bit 2 high */

    return;
}

/*****
/*
/*      POWER_UP()
/*
/*      To turn on the power-down logic on the HEPC2104, the power */
/*      enable bit (bit 2) of the Control register (offset 6), must */
/*      be set low.
/*
/*
*****/
void power_down(void)
{
    outpw(iobase+6,0x00);          /* Set bit 2 low */

    return;
}

```

Example Software for a HEPC2104 with Global FLASH

The HEPC2104 is optionally available with FLASH memory fitted on the Global bus. As such the HEPC2104 can be easily set up to boot from an application loaded into the Global FLASH. This section provides an example using 3L Parallel C tools of how to correctly program the FLASH for self-booting applications.

Although the example is written using 3L Parallel C the main points covered are common to any software tools used to develop a self-booting system with the HEPC2104.

The first program listed below, 'test.c', is a simple test routine that is to be booted on to the processor from the FLASH memory. The program simply sets the IIOF0 pin of the 'C44 to a logic low output, which will result in the IIOF0 LED becoming illuminated as soon as the program has successfully booted.

```

/*****
/
/*                                     TEST.C
*/
/*****
/
#include <c40reg.h>

int main()
{
/* Set the II0F0 Pin to be a logic low output */
reg_clr_iif(0x0000000f);
reg_set_iif(0x00000002);

return(0);
}

```

This simple test routine must be built into a complete 3L image file. The 3L image file can then be programmed into the FLASH memory. The following process requires the use of the 3L ROM Booting Kit to develop a suitable kernel (the file 'hepc2104.krn' below) for a ROM-booting application.

```

c:\> c40c test
c:\> c40cstas test
c:\> config /s test.tsk test.app /a
c:\> copy /b hepc2104.krn + test.app test.img

```

The source file 'test.c' is compiled and linked as a stand-alone task to create the file 'test.tsk'. The task file is then configured into a minimal configuration application file 'test.app'. This application file is then copied with the kernel file to make the final 3L image file, 'test.img'. Note the kernel file must be placed first in the image file, followed by the application file.

Using the 3L ROM Booting Kit

When you load an application using a host system, the 3L command 'tis' first sends a kernel file to the 'C4x network and then sends the application file. However when loading an application from a FLASH memory, or a ROM, there is no host system or server. It follows that the kernel file must be contained in the FLASH memory. This is achieved by combining a ROM kernel and the application into a single image file with the copy command (refer to the previous section for an example).

A ready-made kernel is supplied with the 3L ROM Booting Kit. However, when creating an image file for FLASH booting the HEPC2104 you will need to generate a correct kernel yourself. This is done by describing the hardware in a '.rom' file. The '.rom' file should be a textual description of the information contained in the IDROM of the HEPC2104. As such the values that must be supplied in this file are shown below in an example file 'hepc2104.rom'. (Refer to the section on the IDROM to check the values are right for the hardware you have.)

NOTE: The '.rom' file must not contain an entry for the parameter '_PARAM1'. This parameter details the address at which the FLASH memory is located and must be left out. With this parameter not supplied, the kernel will automatically detect where the boot program is stored, and is guaranteed to find the stored code correctly.

{HEPC2104.ROM}

```

_HOST_GET      &60543210  {boots through logical link 6}
CPU_ID         6          (the processor is a 'C44')
CPU_CLK        32          {this is the value for a 60MHz clock}
MODEL_NO       &000d      {model = HEPC2104}
REV_LVL        &0c        {revision C}
GBASE0         &fffffff   {global memory must not be declared}
GBASE1         &fffffff
LBASE0         &00300000
LBASE1         &fffffff
G_SIZE0        0
G_SIZE1        0
L_SIZE0        &00040000  {this is the value for 1mbyte local}
L_SIZE1        0
F_SIZE         &00000800
WAIT_G0        &0
WAIT_G1        &0
WAIT_L0        &1
WAIT_L1        &0
PWAIT_G0       &0
PWAIT_G1       &0
PWAIT_L0       &1
PWAIT_L1       &0
TIMER0_PERIOD  &00003a98  {this value is based on a 60MHz clock}
TIMER1_PERIOD  &3c
TIMER0_CTRL    &02c2
TIMER1_CTRL    &02c1
GBCR           &1ef78000
LBCR           &1ef78000
_RED_LIGHT     &00000000

```

With this file the required kernel can be built as follows.

```

C:\> gen40 romboot.tsk hepc2104.boo 4 hepc2104.rom
C:\> copy /b hepc2104.boo+kernbase.abs hepc2104.krn

```

The first step runs the 3L boot loader creation program 'gen40' to build the bootstrap file 'hepc2104.boo'. This step requires the supplied bootstrap task 'romboot.tsk', along with the file 'hepc2104.rom' as input.

The second step concatenates the new bootstrap file with the 3L supplied kernel base file 'kernbase.abs'.

Programming the FLASH

With the 3L image file created, the FLASH memory can be programmed with the FLASH booting program. The following source file 'flash104.c' reads the 3L image file and

programs the sectors of the FLASH according to the timing requirements of the memory. (Refer to the 'C4x User Guide for a description of the requirements for booting from external memory).

This example program must follow the programming timing requirements as detailed in the 'The Global FLASH Memory Option' chapter of this manual. In doing so the line

```
#define SECTOR 256
```

must define the correct sector length according to the size of FLASH memory fitted. If the global FLASH is 512 Kbytes, a sector length of 128 is required. If the global FLASH is 1 or 2 Mbytes, a sector length of 256 is required.

The program reads sector length blocks from the 3L image file into a temporary array, before writing the sector data in one continuous sequence to the FLASH memory. This ensures that each access when programming any sector is always followed quickly by the next word access. This is in consideration of the 150us time limit between accesses that triggers the start of the internal programming period.

Note: the first block requires a preceding word with the value 32, to indicate the memory width of the external memory boot-load sequence. Also the next two words which contain the bus control values are hard-coded to ensure the correct values of 1ef78000h are loaded into the FLASH.

In-between the loading of each sector, a 10ms delay is performed to ensure the FLASH has completed the last internal programming period before the next begins.

Finally, when all the sectors are programmed the complete sequence is read back and compared against the 3L image file as a check.

```
/*
/*
/*      FLASH104.C
/*
/*      Example program for loading a 3L application into the FLASH
/*      memory on the global bus of a HEPC2104.
/*
/*      R.Williams      17-6-97
/*
/*
/*
*****/
#include <stdio.h>
#include <stdlib.h>
#include <timer.h>

/* FLASH Sector Size in words */
#define SECTOR 256

/* Pointer to the beginning of the FLASH */
volatile unsigned int *global0 = (unsigned int *)0x80000000;

/* Pointer to the 3L Image file */
FILE *img_fpr;
```

```

int main(int argc, char *argv[])
{
int i,j,k,l,index;
int gsize0,sector,soak;
unsigned int data[512];
unsigned int temp,count;
unsigned int img_length;

printf("\nPress return to load 3L application into memory\n");
getchar();

/* Open the 3L Image file and calculate the file length */
if ((img_fpr = fopen("test.img","rbu")) == NULL){
    printf("Error : fopen failed\n");
    exit(0);
}
if ((temp = fseek(img_fpr,0,SEEK_END)) == -1){
    printf("Error : fseek failed\n");
    exit(0);
}
img_length = ftell(img_fpr);
if ((temp = fseek(img_fpr,0,SEEK_SET)) == -1){
    printf("Error : fseek failed\n");
    exit(0);
}

/*****
/*                               Write the program data to the FLASH                               */
*****/
index = 0;
/* Program first sector */
data[0] = 32;          /* Memory width word */
for (i=1;i<SECTOR;i++) data[i] = getc(img_fpr);
data[1] = 0x1ef78000; /* LBCR Value */
data[2] = 0x1ef78000; /* GBCR Value */
for (i=0;i<SECTOR;i++) global0[index++] = data[i];
/* Wait for 10 milliseconds between programming each sector */
timer_delay(10);

j = (img_length - (SECTOR - 1)) / SECTOR;
k = (img_length - (SECTOR - 1)) % SECTOR;
for (i=0;i<j;i++){
    for (l=0;l<SECTOR;l++) data[l] = getc(img_fpr);
    for (l=0;l<SECTOR;l++) global0[index++] = data[l];
    /* Wait for 10 milliseconds between programming each sector */
    timer_delay(10);
}
for (l=0;l<k;l++) data[l] = getc(img_fpr);
for (l=0;l<k;l++) global0[index++] = data[l];
timer_delay(10);

if ((temp = fseek(img_fpr,0,SEEK_SET)) == -1){
    printf("Error : fseek failed\n");
    exit(0);
}

```

```

/* Check the FLASH contains the correct data */
getc(img_fpr);
getc(img_fpr);
for (i=3;i<(img_length+1);i++){
    if (global0[i] != getc(img_fpr)){
        printf("Error : got %08x (%d)\n",global0[i],i);
    }
}
printf("Done\n");

return(0);
}

```

The FLASH programming application is built as shown below.

```

c:\> c40c flash104
c:\> c40clink flash104

```

This will create the 3L application file 'flash104.app' which can be booted onto the 'C44 of the HEPC2104 by using the 3L server 'tis' as normal.

The default ship state of the HEPC2104 is with the 'BOOT' jumper not fitted. In this state the 'C44 will always boot from a Comport. As such in the default ship state the HEPC2104 can be booted with the 'flash104' application from the host interface using the 3L server, as detailed in previous sections of this manual.

When the 'flash104' application has been successfully run and the FLASH memory programmed with the 'task.img' file, the HEPC2104 is ready to be booted from Global FLASH.

To boot the processor from the FLASH memory, a jumper must first be fitted to the 'BOOT' header. Fitting this jumper causes the 'C44 to look for a boot sequence starting at the beginning of Global memory. With this jumper fitted, the 'C44 will not boot from any of the Comports and accordingly the host interface is no longer required. Instead the boot sequence is taken from the beginning of Global memory which has now been programmed with the 3L image file 'test.img'.

All that is required to cause the processor to boot is the assertion and de-assertion of the TIM-40 system reset. This reset can be driven by any of three sources (refer to the section on TIM-40 Reset). However for embedded self-booting applications where no host interface is present, the 'auto-reset' feature is the most suitable reset source. With a jumper fitted between pins 1 and 4 of the 'RES SEL' header, the TIM-40 reset will be asserted for a short time immediately following power-up of the board. Therefore a short time after the HEPC2104 is first powered up, the 'C44 will begin booting from the Global FLASH.

For the example 3L application that has been covered in this section, this will result in the IIOF0 LED becoming illuminated when the processor has finished booting from the FLASH and has started program execution.

HUNT ENGINEERING have performed testing on its products to ensure that it is possible to comply with the European CE marking directives. The HEPC2104 cannot be CE marked as it is a component in a system, but as long as the following recommendations are followed, a system containing the HEPC2104 could be CE marked.

The immense flexibility of the HUNT ENGINEERING product range means that individual systems should be marked in accordance with the directives after assembly.

1. The host computer or housing in which the HEPC2104 is installed is properly assembled with EMC and LVD in mind and ideally should itself carry the CE mark.

2. Any cabling between boards or peripherals is either entirely inside the case of the host computer, or has been assembled and tested in accordance with the directives. The HEPC2104 inputs are NOT protected against Static discharge, so if the cabling does exit the case, suitable protection must be added elsewhere in the system.

HUNT ENGINEERING are able to perform system integration in accordance with these directives if you are unsure of how to achieve compliance yourself.

Technical support for HUNT ENGINEERING products should first be obtained contacting your local supplier, if you are unsure of details please refer to www.hunteng.co.uk for the list of current re-sellers.

HUNT ENGINEERING technical support can be contacted by emailing support@hunteng.demon.co.uk, calling the direct support telephone number +44 (0)1278 760775, or by calling the general number +44 (0)1278 760188 and choosing the technical support option.