TIM-40 MODULE SPECIFICATION

Including

TMS320C44 Addendum

VERSION 1.01.

Table of Contents

1.THE MODULE CONCEPT

1.1 Introduction

This document describes a cost-effective parallel processing module standard which is intended to improve the features available to parallel processing system designers. The module is completely reusable, allowing different architectures to be experimented with and is available in a range of performance options, allowing the cost of a system to be tailored to the required performance and available budget. The module is intended as a standard hardware building block from which the user may build any conceivable system and onto which he may port any type of software problem.

1.2 Origins of the Module

The module was defined as the result of collaboration with several UK customers whose applications required a module similar to the one described here. These applications varied greatly, covering all aspects of parallel processing - from signal and image processing to parallel supercomputing. Texas Instruments gratefully acknowledges the help received. Without this assistance, the module specification would have been far less flexible and ultimately less useful to many users.

1.3 Module Concept

The TMS320C4x module is a standard hardware platform comprising a processor and some memory or peripherals. While the module is targeted at the TMS320C4x range, it does not preclude the use of other processors. The memory implemented may be of any type, such as fast SRAM, slow DRAM, EPROM, VRAM or cache. No restrictions are placed on the amount of memory, or on its distribution on either the local or global buses. Peripherals may be included to adapt the module to a specific task within a system. For example, an image-processing module might include a frame grabber, an analogue processing board might include ADC and DAC functions, and an audio board might include an AES/EBU interface. The module's physical and electrical characteristics must conform to a specification which details physical and mechanical properties, pin-out, power restrictions and so on, to allow different modules to be plugged together to form a homogeneous system.

1.4 Motherboard Concept

The modules are supported by motherboards, which act as a support environment for the module, providing power supplies, system support functions like clock and reset, and communications paths between modules. Motherboards may be available for a variety of backplanes, such as the PC-AT bus, VME bus, NuBus, FutureBus, Multibus, etc. There is no limitation, as long as the module sites conform to the standard.

A key feature of this concept is that the majority of a user's investment is in the processing and peripheral modules, which make up the system - not the host dependent motherboards. This feature allows systems to be readily duplicated on a wide variety of platforms.

1.5 Summary

Systems may be configured very rapidly using a combination of modules and motherboards. Through this medium, modules may be simply connected to other modules, allowing the user to custom-build a system with no need for hardware development or prototyping.

2. PHYSICAL SPECIFICATIONS

2.1 Introduction

The physical dimensions of the module are perhaps the most important standardisation issue, as they allow modules from different manufacturers to be placed on a standard backplane, possibly from another manufacturer.

This section details manufacturing issues including board area, type and location of connectors, height of module above mother board, etc.

2.2 Dimensions

2.2.1. Board Area (Single Module)

The TIM-40 size offers the best compromise between board area, manufacturability and number mountable on various standard expansion bus boards such as the PC-AT and VME busses.

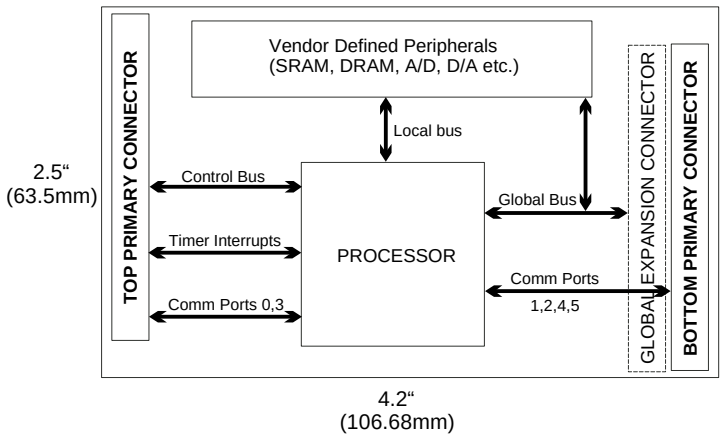

FIGURE 1 - Module showing single-width module from the top.

*Figure I shows **a** single width module from the top. The size of the module allows modules to be stacked vertically along the length of most motherboard formats including the PC format.*

2.2.2. Component Heights

With current surface mount technology, it is possible and desirable to have both sides of a PCB available for component placement. However, the motherboard is capable of carrying components under the module, so some consideration should be given to height restrictions when designing a module to prevent contact. On the upper side of the module, restrictions arise from the height of the top of the module above the main PCB, so again height restrictions must be considered; however, it should be possible for ZIP-packaged DRAM to be used. Figure 2 shows this arrangement.

While no specification is given on the height of components on the top of modules, the designer should consider the maximum permissible heights for various bus formats. The maximum permissible height above the module PCB will also depend on the connector height used. A good guideline would be that a module should be capable of meeting the requirements for single-slot boards when mounted on a motherboard equipped with standard connectors. However, modules may be designed to be used in multi-slot environments.

Recommendations as to component heights relative to connectors used are given in Tables 2, 3, and 4 in the next section. Note that where the global connector is not used, and the space, which it occupies, is used for components, the maximum height of those components should satisfy Table 4.

A MODULE TO MOTHERBOARD HEIGHT	**DEPENDS ON CONNECTOR**
B HEIGHT COMPONENTS - UNDERSIDE OF MODULE	**TABLE 2**
C HEIGHT COMPONENTS - MOTHERBOARD UNDER MODULE	**TABLE 3**
D HEIGHT COMPONENETS - MODULE WITHOUT GLOBAL PIN-OUT	**TABLE 4**
E HEIGHT COMPONENTS - TOP OF MODULE	**OPTIONAL**

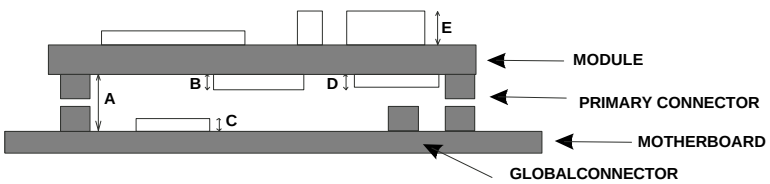

FIGURE 2 - Component heights for the motherboard and module. See referenced tables for dimensions.

2.3 Connectors

2.3.1. Introduction

The TMS320C40 offers the designer six parallel communications ports and two memory busses. Table 1 shows a list of the signals required to support the module's pinout. Note that the full pinout and electrical specification is given in Section 3.

Since some systems may require some form of shared memory an optional secondary connector has been implemented. When present, this connector carries the global memory signals; if not required, the board area used by this connector may be used for components.

SIGNAL GROUP	DESCRIPTION	PINS	CONNECT
Communication Ports	Six TMS320C40 Communication Ports	72	Primary
JTAG Interface	Serial Scan Port for test and debug	7	Primary
Reset	Reset Input	1	Primary
Interrupts	TMS320C40 Interrupt Lines (& IACK, NMI)	5	Primary
Timers	TMS320C40 time control and I/I Lines	2	Primary
Clock Inputs	,Off-module clock source	1	Primary
User Defined Pins	Space lines available to module designers	12	Primary
Clock Outputs	TMS320C40 HI and H3 clock outputs	2	Primary
Data Bus	32-bit data bus for shared memory	32	Secondary
Address Bus	31-bit address bus for shared memory	21	Secondary
Control Bus	Control bus for arbitration of shared memory	17	Secondary

TABLE 1 - Signals required by the Module

2.3.2. Primary Connectors

The primary connectors carry all the signals the module requires to function in a distributed memory environment; in a shared memory environment, they are supplemented by the optional global bus connector (discussed later).

The connector used is the Hirose FX-4 Series from Hirose Electric. This series of connector is available in various heights, from 5mm through to 11mm in 1mm increments. The connector height should be chosen from Tables 2 and 3. Mounting holes are provided on the module and motherboard for securing bolts. The arrangement of connector and bolts is shown in Figure 3. Hirose connectors can be obtained from a variety of electronics distributors. See Appendix 4 for more information.

MODULE Connector Type

FX4C-80P-1.27DSA	< 4mm
FX4C1-80P-1.27DSA (*)	< 5mm
FX4C3-80P-1.27DSA	< 7mm

TABLE 2 - connectors/Component Heights for the Module - Recommended Connector marked (*)

MOTHERBOARD Connector Type

FX4C-80S-1.27DSA (*)	0mm
FX4C2-80S-1.27DSA	< 2mm
FX4C3-80S-1.27DSA	< 3mm

TABLE 3 - Connectors/Component Heights for the Motherboard - Recommended Connector marked (*)

MODULE GLOBAL Connector Type

FX4C-80P-1.27A	0mm
FX4C1-80P-1.27DSA	< 1mm
FX4C3-80P-1.27DSA	< 3mm

TABLE 4 - Connectors/Component Heights for the Global Connector on the Module - Recommended Connector marked (*)

The primary connectors are placed at the top and bottom of the module, as shown in Figure 1. This allows airflow under the module. Keying is provided within the connectors to prevent damage by incorrect insertion.

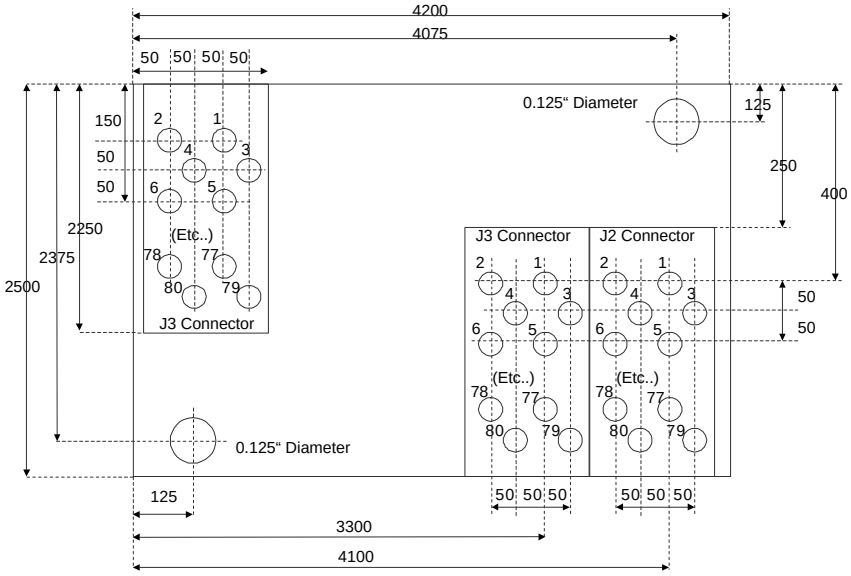

FIGURE 3 - Top view of TIM-40 module showing drill pattern

NOTES TO FIGURE 3

- All dimensions in 0.001” units.
- Connectors on module are of type FX4C1-80P-1.27DSA without flanges and are placed on the underside of the module.
- Holes for J1, J2, and J3 pins should be 0.024” (0.6mm) diameter.
- A silkscreen trianle should be printed in the corner near the J1 connector.
- The two mounting holes should be 0.125” diameter, suitable for M3 fixing bolts.
- No tracks or components should be placed within 0.125” of the centre of the mounting holes on outer layers of the board.

2.3.3. Optional Global Bus Connector

In the case where the global memory bus is needed off the module, an optional secondary connector is provided.

The exact location of the connector is as shown previously in Figure 2. Note that all three connectors (if fitted) are parallel, allowing the maximum possible airflow under the module.

Note that where the global connector is not used, and the space, which it occupies, is used for components, the maximum height of those components on the underside of the module should satisfy Table 4. The motherboard height restrictions of Table 3 apply to the components under the global connector.

2.4. Multiple Width Modules

An important requirement for a module is the ability to implement double, triple or even quadruple width boards, allowing a designer to work with different board sizes for different functions. This section details the physical interrelationships of modules on the motherboard for multiwidth modules. The spacing of module sites on the motherboard is controlled by the specification to allow large modules to span multiple sites. The arrangements for mounting modules in this way are shown in Figure 4.

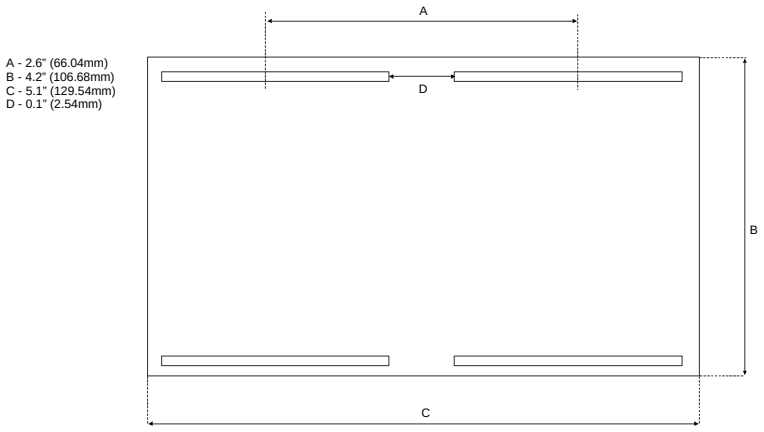

Figure 4 – Connector Relationship for Multiwidth Modules

Key points here are that all modules on the board have the same orientation, so that all secondary connectors are adjacent; this allows shared memory systems to implement the required PCB tracking for the memory buses more easily. It is also worth noting

that while the PC format may support triple or quadruple size modules, other formats such as VME may only support double width modules. The contents of a multiple width module are left totally to the designer's discretion. Like its single width partner, the multiple width module can carry any number of processors, any amount of memory, or any number or type of peripherals.

A double width module will be equipped with two sets of connectors to provide it with sufficient stability and rigidity. The additional connectors can be used to provide additional power supplies, or additional communications port routes from the CPU to the motherboard. It is recommended that a multiple width module implements the processor to the left of the module, (with J1 positioned at the top), and uses the additional right hand connectors for power supply only. This recommendation allows some standardisation of modules, allowing standard motherboard routing patterns to be implemented more easily.

It is also recommended that unused communications ports should, where possible, be routed through the module; for example, if a double width module uses six ports, the other six should be connected from outputs to inputs. This structure allows communications links to run across motherboard sites occupied by dummy modules.

2.5. Multiple Processor Configuration

It is important in some applications that software running on a system be implementation independent. Care must be taken in two areas to ensure implementation independence:

- In the case of a motherboard carrying a processor (an intelligent motherboard), software should not be able to distinguish the processor on the motherboard from other processors in the system;
- In the case of a module with more than one processor, software should not be able to determine that these processors are not on separate modules.

In the case of the motherboard, the solution is for the processor on the motherboard to be equipped with an ID ROM, and for it to communicate with the rest of the network as if it were a module.

In the second case, the multi-processor module must logically implement an ID ROM per processor, and again communicate with the rest of the network as if it were multiple modules. An important point here is that the logical ID ROM supplied to each processor may be the same device, with sharing implemented in hardware.

For a processor to communicate as part of the network requires only that a link be connected to another processor in the network. This is a requirement for being part of the network, so is implicit.

2.6. Redundant Communications Ports/Connectors

It is left to the user to decide whether a module should use all of its links, or keep some links local. This is of little concern to the single width/single C40 module, where the processor has six links, matching the module connectivity. However, other combinations of processor and module may result in some mismatch.

As an example, consider an extreme - the single width/quadruple C40 module. The four processors implement 24 links, some of which will be used to communicate between the processors. Where there are spare links, their use is left to the designer's discretion.

At the other extreme, consider the case of the double width module with a single C40. In this case, 12 link connectors are available, while the processor implements only six communications ports.

It is also recommended that unused communications ports should, where possible, be routed through the module; for example, if a double width module uses six ports, the other six should be connected from outputs to inputs. These measures are necessary to ensure that an interconnected network can still be achieved when using this module.

The most extreme case of this is where a peripheral module requires only a single link. In this case, it is recommended that link 0 is used, or where two links are required, link 0 and link 3. This provides a guideline for link routing on motherboards which do not provide link switching facilities.

2.7. Redundant JTAG connections

It is important that care should be taken that the JTAG chain is unbroken across a motherboard. Two cases may cause problems

- double width modules o module without JTAG devices
- module without JTAG devices

The bypass facility is implemented through a simple wire link on the module itself. This is conceptually simple, but note that where a slot is bypassed in this way the module MUST assert the SENSE and SENSE* lines for that slot; if this is not done, any motherboard bypass circuitry will be enabled with its inputs and outputs shorted together.

Note that double width module designers must treat JTAG as having two distinct ports onto the board. JTAG hardware may be distributed between the two JTAG ports as required; however, it is not permissible for the JTAG input signals of one port to be shorted across to the JTAG signals of the other port, as it would theoretically be possible for the two JTAG ports to exist on separate test chains.

3. ELECTRICAL SPECIFICATIONS

3.1 Introduction

One of the most important aspects in the specification of the module is the definition of signalling requirements, levels and timing parameters. As this module is designed primarily for use with the TMS320C40 parallel processor, many of the timing details are referred to the data sheets for that device. However, many of the global bus connections may be used for support of peripheral devices, and are defined here in more detail.

3.2 Signal Requirements

The module requires a large number of signals to give the user full access to the TMS320C40's powerful communications capabilities. Many of these lines require very little definition, as they are simple clocks or strobes, or are defined in detail in the TMS320C40's User's Guide. The minimal set of lines required to implement a TMS320C40 distributed memory system are included on the two primary connectors, while the global memory bus, required only for sophisticated shared memory architectures, is included on the optional secondary connector.

FUNCTIONAL GROUP	DESCRIPTION	NUMBER
Communication Ports	TMS320C40 Communication Ports	72
JTAG Interface	Serial Debug Port	7
Reset	Reset Control Lines	2
Interrupts	TMS320C40 Interrupt Lines (incl. NMI)	5
Timers	TMS320C40 Timer I/O	2
User Defined Pins	Space lines available to module designers	12
Power (+5V)	1.5A 5V rail	15
Power (+12V)	250mA max +12 rail	2
Power (-12V)	250mA max -12 rail	2
Ground	2A Ground	40

Table 5 - Standard Module Pin Assignments

Table 5 contains a list of all the primary connector signals, while Table 6 lists all the secondary connector signals. A full pin-out for the connectors is given in the Appendices at the end of this document.

FUNCTIONAL GROUP	DESCRIPTION	NUMBER
Data Bus	32-bit Data Bus	32
Address Bus	31-hit Address Bus	31
Arbitration Bus	Arbitration Signals	9
Control Bus	Control Bus with strobes, etc.	8

Table 6 – Global Memory Pin Assignment

3.3 Power Recommendations

Power is supplied to the module through multiple pins to ensure a clean, low impedance power supply is available. Power rails of +5V, +12V A -12V are available, and should be supplied by all motherboards. It is recommended that as a minimum, a single large tantalum capacitor is used to decouple these power rails where they enter the module, with individual devices decoupled in the normal way. Specialist modules may require more attention to power supplies than this, for example, for high precision ADCs etc.

It is important that all power and ground pins be connected together on the module, as an individual pin may be used to act as a presence detect for the module.

3.4 Identification ROM

A small ROM is included in the module to allow storage of serial numbers, setup data and so on. More details of how the ROM is used and accessed by the standard software is given in section 4.

Two versions of the ROM are available; a 4-bit ROM, and a serial ROM. In either case, TCLK0 is driven high to signal ID ROM access, and the ROM should be decoded on STRB0. If there is any other memory attached to STRB0, it must be disabled while the ID ROM is accessed. The standard software to access the ROM will run from internal memory, before any external memory has been used or initialised. This allows some simplification of the hardware.

3.5 CONFIG Line

The module is equipped with a CONFIG pin. This is an open collector output from the module, which is asserted by software on the board during the first stage of bootstrapping. It remains asserted throughout the bootstrap; the final bootstrap code should de-assert

the line before passing control to the application program.

As this line is asserted during the network exploration stages of bootstrapping, it can be used to protect peripherals from unwanted data packets. This feature may be desirable where a peripheral, or circuitry connected to it, may be damaged by exploratory packets issued by a worm program. An example of this could be an audio DAC, where worm packets could be driven out to an audio amplifier.

The standard worm program uses IIOF3 for this function. The line is asserted at the start of bootstrap, as described above. Figure 5 shows an example schematic which could be used to implement the CONFIG function.

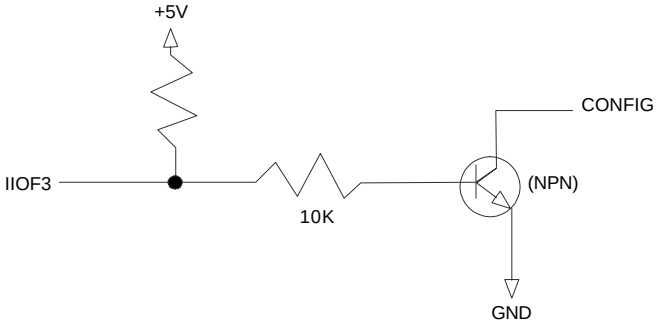

FIGURE *5* – Possible Config line schematics

3.6 Clock Generation

3.6.1. Distributed Memory Case

Modules for distributed memory systems should require as little support as possible, and should therefore contain their own oscillator. This, combined with the asynchronous nature of the TMS320C40's communications ports, allows them to directly control their own speed. This allows fast CPU's to run to the best of their ability, rather than being tied to the lowest common denominator clock frequency.

Note that in this case, the widest range of clock frequencies available is limited by the TMS320C40's communication port. This can only operate correctly between ports with a factor of less than two difference between clock rates - so a 50MHz C40 processor could communicate with any TMS320C40 device with a clock between 25MHz and 100MHz.

A CLKIN line is available on the primary connectors, so a module may choose to take its clock from the motherboard. This approach eliminates the need for an oscillator on the module, but forces the CPU to run at a rate controlled by the motherboard.

3.6.2. Shared Memory Case

Shared memory systems are a slightly more complex case than that just described, in that shared memory designs are usually synchronous. To provide for such systems, the processor's Hl and H3 clocks are available to the motherboard, while the primary connector carries a clock input line. A motherboard should be capable of supplying a range of low-skew clocks to the modules, with some form of user selection for each individual module.

Using this system, all the modules on a motherboard sharing a memory pool would be clocked from one low skew source. Assuming that the shared memory is accessible to only one motherboard, there is no need for motherboards to have low skew between their clocks.

3.7 RESET Generation

The modules are equipped with a single RESET line - RESET* This, in conjunction with reset control logic on each motherboard, allows a simple control structure to be implemented, where individual processors, groups or threads of processors may be reset independently of the main network.

The timing of the RESET* line should satisfy the minimum RESET* period specified for the TMS320C40. A good minimum for this would be 10ms, which, while significantly more than required by the processor, will reliably reset all common VLSI devices.

Sophisticated control of RESET is possible through the module's JTAG port, allowing any processor to be selected from the network and reset. In conjunction with the other facilities provided by the JTAG port, sophisticated reset structures may be built up.

3.8 Boot Control

It is anticipated that most modules will bootstrap from links, and will therefore power up with IIOF(3- 0) all pulled high. This feature allows a network to be started from a single communications port. However, the TMS320C4x has a variety of bootstrap modes which may be more appropriate under certain circumstances - for example, a free-standing network may have to boot from ROM, or a network may require one processor to bootstrap from a byte wide ROM before accessing a SCSI disk. This specification does not restrict these possibilities; however, the ability to bootstrap from link is useful for debugging ROM code during development.

Bootstrapping requires that the on-chip ROM is enabled and that IIOF(3-0) are held in a known state as the device comes out of reset. Full details of the bootstrap loader are given in Section 13 of the TMS320C40 User's Guide, but booting from link may be easily accomplished by using pull-up resistors on IIOF(3-0) and on ROMEN.

Note that selecting the start mode of a module is that module's own responsibility. The motherboard will tristate all mode control lines

during the reset phase of bootstrapping.

3.9 Unused Links / Signal Conditioning

If the module is intended to run links outside a single rack, it is recommended that some form of buffering is used. This is described in the TMS320C40 User's Guide, and is outside the scope of this document.

Each C40 has six links. When a network is built from standard modules, it is unlikely that every link will be used, due to physical restrictions on the motherboard, or the requirement to leave hooks for expandability.

In this case, some thought should be given to the treatment of the data lines of these links.. The C40 User's Guide suggests that 20K pullups be used on those ports which are configured as inputs after reset. These should be placed on the motherboard. Where a link is taken to a board connector, it is only necessary to provide pullups on those lines that are inputs to the C40 at RESET.

3.10. Presence Detection (SENSE function)

To allow motherboards to determine whether or not a module is present, a "SENSE" function is included in the module specification. This indicates directly the presence (or absence) of a module.

Two lines are implemented; SENSE and SENSE*. On the module, SENSE should be provided with a low impedance connection to the 5V rail, while SENSE* is grounded. The module should effectively treat these signals in exactly the same manner as power supply pins.

Motherboard hardware to implement module detection is then limited to one resistor; either a pulldown on SENSE, or pullup on SENSE*. The resulting signals can be used to implement pass-through functions for the JTAG daisy chain, or to connect some of the site's communications ports together.

A large value resistor is recommended for this function. Note that this part of the specification is compulsory for modules, but is optional for motherboards. Should a motherboard designer choose not to implement the SENSE function, it is recommended that the two signal lines be used as power and ground feeds to the module, giving the module improved power and ground rails.

A multiwidth module should only assert the SENSE lines on the module site which uses the primary connectors. If a multiwidth modules does not use a module site for anything other than power supply and ground then it should not assert the SENSE lines.

4. IDENTIFICATION ROM FORMAT

4.1 Introduction

In order to assist software to determine the type of hardware on which it is running, and to ease the task of locating peripherals like SCSI or graphics interfaces within the network, a small ID ROM is included in the module. The ROM is located at a predefined address and is accessed by the TMS320C40 during bootstrap. It is also used to hold hardware dependent details like set-up values for bus interface controllers and interrupt mask registers etc.

4.2 ROM Type

Three different ROM types are available for the ROM driver software. These are differentiated according to the bootstrap method:

1) Communications Bootstrap

In this case, the state of the TCLK0 pin is read by the ROM driver. If this pin is low, the ROM is assumed to be a 4-bit wide device, located on the local memory bus, on LSTRB0. If it is high, a serial access ROM is assumed, accessed in the same way as the 4-bit device.

2) Bootstrap from ROM

In the case of ROM bootstrap, the ROM driver will default to loading the ID table from the bootstrap ROM. The bootstrap ROM can be of any type and width, as discussed in the TMS320C40's User's Guide. A point worth noting with this format is that the ID block in the ROM cannot be the first block, as it will then be interpreted as executable code.

In all of these cases, the ROM is assumed to be stored with the least significant bits first. In the case of a 16-bit ROM storing 0x12345678, this would mean the first 16-bit word will contain 0x5678, while the second word contains 0x1234; this example could be extended to a 4-bit ROM, where the first few words would contain 0x8, 0x7, 0x6, etc. Finally, for a serial ROM, the LSB would occupy the first bit of the ROM.

4.3 ID Data Loaded

Most of the ID ROM data is stored in a packed structure located within the first few words of on- chip memory. For details of that structure, see section 4.6.; the remaining data is used for system configuration, initialising timers and bus interfaces. The rest of the ROM may be used for autoinitialising registers in peripherals. This data will be loaded to specific addresses in the processor's memory map for example, status registers or control registers in external peripherals. The following notes explain the contents of the ROM, aid the uses of each codeword.

PROCESSOR	ID CODE
TMS320C40	0
TMS320C30	1
TMS320C31	2
TMS320C3x	3
TMS34010	4
TMS34020	5

TABLE 7 - Processor ID Codes

ID_SIZE - This 32-bit code contains the number of words that will be transferred by the boot loader into RAM. It excludes both itself and the destination address which follows it, as these words would not be transferred by the boot loader. More details on the boot loader may be obtained from the TMS320C4x User's Guide.

RAM_ADDRESS - This is the destination address for the ID ROM table. This should be the base of on-chip memory, providing a standard buffer for the first phase of bootstrapping; however, the program reading the ID ROM is free to place it in any convenient memory buffer.

TABLE_SIZE – This 32 bit code gives the number of 32-bit words held in the ID ROM, excluding all autoinitialisation information. This code is guaranteed never to exceed 32, giving the maximum buffer size required to hold the ID ROM information. It excludes the first two words of the ID ROM, which are control words for the bootloader and hence need not be transferred into RAM.

MAN_ID – This 16 bit code is a unique code assigned to a specific manufacturer. The code is unique to a manufacturer. No manufacturer may use more than one code, nor should a code be used before it has been assigned by the controlling committee. Assigning a number is just a formality, but must be controlled to ensure codes remain unique.

CPU_ID - This is an 8-bit code identifying the main processor on the module. As this document details a module aimed at the TMS320C40, it is the most likely CPU. However, it is possible to use the module standard with other processors, from Texas Instruments or from other suppliers. It is also possible to have a module without a processor. Again, if a CPU code is not available for the chosen CPU, a number must be assigned by the controlling committee. Table 7 lists CPU codes currently assigned.

CPU_CLK - This 8-bit code details the instruction cycle time of the processor. A code of zero represents 1 ns cycle time, while each increment represents a further nanosecond - as an example, a 40ns TMS320C40 would have a CPU CLK code of 39. Cycle times of between 1 and 256 ns are offered using this approach.

Note that on the TMS320C40, the instruction cycle time is identical to the Hl clock rate. A 50MHz C40 will have an Hl clock of 25MHz, or 40ns cycle.

MODEL_NO - A 16-bit code identifying the module within a manufacturer's range. This, combined with the manufacturer's ID code also contained within the ROM, can be used to uniquely identify the module and its function. While not essential, it is advisable to inform the controlling committee which codes have been used; this will allow operating systems to determine the type of modules present within a system accurately.

REV_LVL - This 8-bit code represents the manufacturer's revision level, for example, Rev 1.0, 1. 1 etc. This data is at the discretion of the manufacturer.

RESERVED – These 8 bits pad the area used for non-word quantities to a word boundary. They are reserved for future use and must read as zeros.

GBASE0, LBASE0, GBASE1, LBASE1 – These 32-bit addresses are pointers to the start of memory on each external memory strobe. Nominally, this specification gives guidelines for memory location, but these codes increase the flexibility available to the hardware designer. If memory is contiguous between the two strobes on a memory interface, the two banks should still quote separate base addresses.

If no memory is implemented on a strobe, the base address should be set to 0xFFFFFFFF, while size, waitstates and page miss waitstates (PWAlT) should all be zero, indicating no memory present. The only exception to this rule is where a board is shipped with an indeterminate amount of memory on a strobe – for example, a module shipped with empty sockets for memory. In this case, the base address should be set to the appropriate value for the memory, but SIZE should be set to zero. This should request automatic size detection from the bootloader software, resulting in the replacement of the S1ZE code with a valid code.

GSIZE0, LSIZE0, GSIZEl, LSIZE1 – These 32-bit counts give the size of the off-chip memories, in words. If added to the equivalent base address and decremented once, they will give the address of the last word in that memory bank. If memory is contiguous between strobes, the size of each bank should be quoted separately. If memory addressed by a strobe is non-contiguous from the 'associated memory base address, the size of the first bank of memory addressed by that strobe should be given. This approach means that a special driver would have to be used to access the non-contiguous block, so this type of memory structure is not recommended.

If no memory is implemented on a strobe, the base address should be set to 0xFFFFFFFF, while size, waitstates and page miss waitstates (PWAIT) should al**l** be zero, indicating no memory present. The only exception to this rule is where a board is shipped with an indeterminate amount of memory on a strobe – for example, a module shipped with empty sockets for memory. In this case, the base address should be set to the appropriate value for the memory, but SIZE should be set to zero. This should request automatic size detection from the bootloader software, resulting in the replacement of the SIZE code with a valid code.

FSIZE – This 32-bit count specifies the size of the contiguous fast memory pool, including the on-chip memory. In many cases, this will simply comprise the on-chip RAM. However, if the location guidelines given in this specification are followed, the on-chip memories may be extended with zero wait state SRAM externally. In this case, that memory should be included in this count.

If there is no external fast memory, only the on-chip memory should be included in this count.

WAIT_G0, WAIT_G1, WAIT_L0, WAIT_L1 – These 4-bit codes give the number of CPU CLK cycles associated with a read of each bank of memory in the system. Page mode memories should give the in-page access time here. Note that this code is for memory, and does not apply to peripherals. Where multiple banks of memory are implemented on a strobe, this timing should be set to 0xF, to indicate indeterminate. Note that a zero wait state read requires a single cycle.

If no memory is implemented on a strobe, the base address should be set to 0xFFFFFFFF, while size, waitstates and page miss waitstates (PWAlT) should all be zero, indicating no memory present.

PWAIT_G0, PWAIT_G1, PWAIT_L0, PWAIT_L1 – These 4-bit codes give the number of CPU CLK cycles associated with a read of each bank of memory in the system, assuming a page boundary has been crossed. Where a non-pagemode device is in use, this should be set to the standard access time for the device. Note that this code is for memory, and does not apply to peripherals. Where multiple banks of memory are implemented on a strobe, this timing should be set to 0xF, to indicate indeterminate. Note that a zero wait state read requires a single cycle.

If no memory is implemented on a strobe, the base address should be set to 0xFFFFFFFF, while size, waitstates and page miss waitstates (PWAlT) should all be zero, indicating no memory present.

TIMER0_PERIOD – This 32-bit value is intended to be loaded into the C40's timer 0 period register, and should generate a 1ms period if the timer is running in pulse mode. (This timer mode allows the highest resolution, as it eliminates a divide by two in the timer clock chain). The goal is to provide the user with easy access to a high-accuracy clock which may be used for scheduling interrupts. Given this count, any other period should be obtainable by scaling – note that the timers use an offset count scheme.

TIMER1_PERIOD – This 32-bit value, in conjunction with TIMER1 CTRL, controls the operation of the C40's second timer. This timer's use is left very much up to the module designer, so there are no restrictions as to the setup used. If a module does not use this timer, this value should be zero to allow software to determine its availability.

TIMER0_CTRL – This 16-bit count should be zero-Bled to 32-bits before loading into timer 0's control register. This code should place the timer in pulse mode, running from internal clock, while leaving the external timer pin as general purpose I/O.

TIMER1_CTRL – This 16-bit code should be MSB padded with zeros before loading into timer 1's control register. See notes on TIMER1 PERIOD for more details. This value should be zero if the module does not use timer 1.

GBCR, LBCR – These 32-bit codes axe used to directly initialise the C40's bus interfaces, setting up details like how waitstates are generated, number of software waitstates to be used, size of page boundaries, and how the various control strobes interact.

AINIT_SIZE – This is the total size of the autoinitialisation table following the compulsory ID information. It has an absolute maximum of 222 words; this can be regarded as the maximum buffer size required to hold the autoinitialisation table. Where there is no autoinitialisation data, the ID ROM should still terminate with a zero word for consistency with the ROM bootstrap program. Therefore, the minimum value of AINIT SIZE is 1 – it cannot be zero.

'Note that the ROM drivers will first transfer this entire table into on-chip RAM, in a packed structure. Thus, enough space should be left at the base of the on-chip memories to accommodate the entire structure. Either TABLE SIZE or ID SIZE can be used to

determine the amount of memory required. The bootstrap ROM driver will only transfer those words that are destined for the table; data used for initialisation of control registers will be transferred directly to those registers.

Also, note that there is no requirement for a module to maintain a copy of the ID ROM in on- chip memory once booted. It would be possible to copy the table into slower off-chip memory, or even wipe it completely once the necessary information has been extracted.

4.4. Assignment of Manufacturer's ID codes

ID ROM codes are assigned by the TM-40 co-ordinator. The address and contact numbers are given in the Appendices at the end of this document, along with an application form which may be mailed or faxed to the co-ordinator.

It is important that you should supply as much information as possible on the application form. This information is important in that it allows users of the standard to be updated as new releases of the documentation become available.

4.5. Reserved Manufacturers ID ROM Codes

Two manufacturer's ID codes are reserved for use by the specification. They are intended for development work, and for software signalling; they must never be used on production boards. The reserved codes are:

0x0000 - guaranteed never used 0xFFFF - available for use during prototyping. Must not be used for production boards.

Neither of these codes must ever be used on a production board.

4.6. ID ROM Format

The format for the ID ROM is defined to allow the ROM bootloader to transfer the ID ROM table to RAM automatically. This occurs as part of the normal boot sequence on a ROM-boot device, but would have to be explicitly called on a communications-boot device. It is recommended that in the case of the communications-boot, this is done as one of the earliest tasks in the system as initialisation of the bus controllers may render the ID ROM inaccessible. Listing 2, 3 shows a TMS320C40 assembly listing example of the format used.

4.7. Control Register Autoinitialisation from ID ROM

It may be desirable to have some mechanism for initialising control registers for external hardware. This can be handled simply using the standard ROM bootstrap code. During boot, this will read information from the ROM in triplets of 32-bit words. The first word is interpreted as a 32-bit block size, the second as a 32-bit address, while data from the third word onwards is written to that address. This provides a simple means of register initialisation that is compatible with the standard bootstrap code.

Where a serial ID ROM is used, these 32-bit triplets are transferred to on-chip memory before being loaded into their destination addresses. This approach can also be used with the boot ID ROM, and has the advantage that all data is transferred out of the ID ROM before the system is configured.

4.8 ID ROM Boot Sequence

The sequence of events which result in the ID ROM being loaded and transferred to control registers etc. varies depending on the bootstrap source. What follows is a generic description:

1) Processor boots. In the case of the ROM boot, the ID tab]e is transferred to the base of on- chip memory. In the case of the comms port boot, the ROM driver is downloaded.

2) A processor booted by link runs the ROM driver. The ROM-booted processor skips this stage.

3) Processor transfers internal autoinitialisation data to destination addresses. This covers bus control registers, timers etc.

4)Processor then looks for any autoinitialisation blocks for additional initialisation, and transfers these. In the ROM boot case, this job may already have been performed by the bootstrap code itself.

4.9. Communications Port Bootstrap

```
************************************************************************************************************************************************************
ID_ROM.asm

File contents in format suitable for either the TMS320C40 bootloader or the TIM-40 ROM driver
************************************************************************************************************************************************************

ID size         .word   ID_start-ID_end         ;size of ID block to load
RAM_Address     .word   02ff800h                ;start of on-chip RAM

;start of fixed information for ID ROM. .FILED is used to create a packed structure
ID_start
TABLE_SIZE      .word   ID_start-ID_end         ;size of ID block to be loaded
MAN_ID          .field  MANUFACTURER,16         ;manufacturer ID code
CPU_ID          .field  PROCESSOR,8             ;processor ID code
```

```
CPU_CLK         .field      CLOCK,8             ;processor clock rate
MODEL_NO        .field      MODEL,16            ;manufacturer model number
REV_LVL         .field      REVISION,8          ;revision level
RESERVED        .field      ZEROS,8             ;reserved field, read as zero
GBASE0          .word       GBASE_0             ;base of global strobe 0
GBASE1          .word       GBASE_1             ;base of global strobe 1
LBASE0          .word       GBASE_0             ;base of local strobe 0
LBASE1          .word       GBASE_1             ;base of local strobe 1
GSIZE0          .word       GSIZE_0             ;size of global memory, strobe 0
GSIZE1          .word       GSIZE_1             ;size of global memory, strobe 1
LSIZE0          .word       GSIZE_0             ;size of local memory, strobe 0
LSIZE1          .word       GSIZE_1             ;size of local memory, strobe 1
FSIZE           .word       FSIZE_RAM           ;size of fast RAM pool
WAIT_G0         .field      WAIT_G_0,4          ;no cycles to read GSTRB0
WAIT_G1         .field      WAIT_G_1,4          ;no cycles to read GSTRB1
WAIT_L0         .field      WAIT_L_0,4          ;no cycles toread LSTRB0
WAIT_L1         .field      WAIT_L_1,4          ;no cycles to read LSTRB1
PWAIT_G0        .field      PWAIT_G_0,4         ;no cycles to open page, GSTRB0
PWAIT_G1        .field      PWAIT_G_1,4         ;no cycles to open page, GSTRB1
PWAIT_L0        .field      PWAIT_L_0,4         ;no cycles to open page, LSTRB0
PWAIT_L1        .field      PWAIT_L_1,4         ;no cycles to open page, LSTRB1
TIMER0_PERIOD   .word       TIMER0_period       ;period count for 1ms timer
TIMER1_PERIOD   .word       TIMER1_period       ;period for timer 1 (H/W defined)
TIMER0_CNTL     .field      TIMER0_CNTL.16      ;16-bits for timer 0 control reg
TIMER1_CNTL     .field      TIMER1_CNTL,16      ;16 bits for timer 1 control reg
GBCR            .word       G_BCR               ;global bus control register
LBCR            .word       L_BCR               ;local bus control register
AINIT_SIZE      .word       AI_start-AI_end     ;total size of autoinitilisation reg
ID_end

;End of Statutory ID ROM Information - Next section is date for autoinitialisation
AI_start
B_SIZE1         .word       SIZEOFBLOCK         ;size of first autoinitilisation block
B_ADDR1         .word       ADDR_OF_BLOCK       ;destination address
B_DATA          .word       1,2,3,4,5           autoinitilisation data
B_SIZEn         .word       0                   ;block size zero to end autoinitilisation
AI_end
```

In the case of a communications port bootstrap, the ID table is loaded by a driver downloaded to the processor as part of the boot sequence. There are a couple of extra requirements for this type of ID ROM, covered in the following sections; these deal with the special needs of this type of ID ROM.

4.9.1. Protection of Writable ROMs

It is recommended that protection circuitry be used to enable the ID ROM if a writable technology is used. The ROM driver will drive the TCLK0 pin high while accessing the ROM, so this line could be tied to the device's decode, ensuring that it cannot be written to by an errant program. The same line could also be used to disable other memory devices attached to the same memory strobe. This note applies ONLY to the case of the communications port bootstrap.

4.9.2. Recall of EEPROM Arrays

Several of the EEPROM devices likely to be used for the ID ROM function are based on a combination of RAM and EEPROM. These devices require a pulse on a "RECALL" pin to transfer the non-volatile contents into RAM. This is catered for by the standard software driver's actions after RESET; TCLK0 will be held low for a minimum of 200us after RESET until the ROM driver starts a recall sequence, then driven high for around 200us, low again for 200us, then high for accesses. This line can then be tied to the "RECALL" pin on many small EEPROMs. This note applies ONLY to communications bootstrap.

4.9.3. Location of Serial and 4-bit ID ROMs

The serial and 4-bit wide ID ROMs reside by default in the TMS320C40's local memory map, and are accessed using LSTRB0. When accessing the ID ROM after a communications port bootstrap, the C40 configures its bus so that LSTRB0 is valid for the entire local memory map, disabling all devices attached to LSTRB1, then accesses the ROM from address 0x70000000. TCLK0 will be high during these accesses, so can be used to disable anything else connected to LSTRB0.

I/O will be through software; a standard software driver for the ROM, which all modules must conform to, is available from the TMS320C40 Consortium.

4.9.4. ROM Driver for Serial/4-bit ID ROMs

It is anticipated that the standard ROM driver be downloaded as part of the process of starting a network. This driver is available from the TMS320C40 Consortium.

4.9.5. Accessing Serial/4-bit ID ROMs

The following information is intended for those developing code to access the ID ROM, and also as a guideline for hardware designers developing modules.

4.9.5.1. Bus Controller Set-up during ID ROM Access

The ROM driver should use the C40's power on reset values for the bus controllers when accessing the ID ROM. This provides a standard reference point for all board builders and software writers to adhere to.

4.9.5.2. Accessing the ROM

Many serial ROMs do not support true random accesses, but provide data in a purely sequential mode. Any software written to access the ID ROM should accept this constraint; if data is not used in the order it is read from the ROM, it should be held in RAM or registers until it is used. It may not be possible to return to a lower address without resetting the ROM device.

This constraint may also apply to some 4-bit ROMs. Code that accesses the ID ROM should do so with sequential addresses (0x70000000, 0x70000001,.) and must not assume any ability to skip addresses or to 'rewind' the ROM.

4.9.5.3. Unused Bits

On reading the ROM, the processor will receive a 32-bit word with either one or four bits of ID data embedded into the LSBs. No assumptions can be made regarding the state of the other bits. They will be required to be masked out before the data is used.

4.9.5.4. Resetting the ID ROM

It should be possible to reset the ID ROM by following the sequence described in section 6.8.2, "Recall of EEPROM Arrays". Sequential access devices can use this sequence to reset their address counters; true random access devices may ignore this sequence (other than for control purposes).

4.9.5.5. Maximum Access Rate

Many commonly available serial ROMs are intended for use in low speed serial bus systems, with data clocked in by a relatively slow clock – typically 1MHz or less. In order to cater for these devices, the ROM driver should access the ID ROM at most once every 2us, giving a read rate of 0.5MHz. Any other concessions for such devices must be catered for in hardware.

4.10. ROM Bootstrap

Where a module bootstraps from ROM, the ID table load is performed by the processor's own bootstrap code. This is implemented through the standard driver. The following notes detail some of the special needs of this type of ID ROM.

4.10.1. Protection of Writable ROMs

As the ROM bootstrap model uses the standard C40 bootstrap code, no protection is implemented. However, this is not seen as a major drawback; it is likely that the ROM used in this case would be a non-writable technology.

4.10.2. Recall of EEPROM Arrays

Again because ROM bootstrap uses the standard C40 bootstrap code, no Recall facility is implemented. Once again, this is not seen as a problem, as a boot ROM which must be recalled from software is of very limited use.

4.10.3 Location of boot ID ROMs

Any standard boot ROM location may be used; again, this is a direct result of the use of the standard bootstrap code.

4.10.4. ROM Driver for boot ID ROMs

The primary ROM driver used for boot ID ROMs is the standard C40 bootstrap program. This provides all the functionality required of the ROM driver. Secondary loading functions are handled by the TIM-40 ROM driver, available from the TMS320C40 Consortium.

5. MOTHERBOARD RECOMMENDATIONS

The module concept described here is a blend of hardware and software. For the module to be successful, it is essential that modules be supported by motherboards, and that the two basic components should be mergeable into a single vendor-independent computing system. The hardware of the module itself is reasonably straightforward, so few software issues are expected. However, it is equally important that the motherboard be similarly well defined, allowing standard host server software to be used across a variety of platforms.

This section details the requirements of the motherboards ranging from single slot motherboards to multi-motherboard systems.

5.1 Basic Motherboard Concepts

Within the scope of this section of the TIM-40 specification, we are considering motherboards for TIM-40 modules which are hosted on some other platform - for example, a Sun SPARCstation, or an IBM PC. It is assumed that the host machine will be running some form of server software, providing the network with the ability to access disks, graphics screens, or networks.

Many applications will use this configuration. Many more will use systems where the C40 network itself provides many of these services, but are still hosted on some other machine. Many more again will be self-hosted, where the C40 system runs entirely stand-alone. This section of the specification is intended to also address the needs of these systems.

There are three basic requirements of the motherboard system; these are:

1) Provide a basic operating environment in which modules can operate, and communicate with other modules.

2)Provide a communications environment between the module network and the host machine, at three levels:

- data communications, allowing message transfer between modules and the host;
- debug interface, allowing the host total control over the operation of the network;
- subsystem control, allowing the host to reset individual modules should the need arise.

3) Provide scalability within this environment, so that large networks can be constructed from multiple motherboards.

By meeting all three requirements, it is possible to implement a system which appears to the host as one homogeneous network, where in fact it may be made up of multiple motherboards, some physically within the host machine, and some external.

Extreme examples of this might range from a network of 16 processors, mounted on four motherboards, all located within the case of an IBM PC, to the case of a distributed 16 processor network, where the processors communicate over optical links and may be a considerable distance apart. Such a system may be hosted on a Sun SPARCstation, where the card mounted in the workstation simply provides an interface to the network.

To maintain a similar interface between these two systems, it is essential that a network has a single access point to the host machine, and that all system services may be extended to other motherboards. This is achieved by designating one motherboard to be the 'master', and using this *m* a single entry point into the network.

5.2 Motherboard Subsets

It may be felt that for some specialist applications the subsystem provided by the motherboard is excessive. In these cases, it is quite permissible for a system to be constructed using TIM-40 modules with a reduced motherboard specification. Such an exercise would probably result in lower system cost; however, it would also prevent much standard software from running on the system, and may compromise the operation of some hardware modules.

In view of these issues, it is recommended that any motherboard which is intended to be used freely within the TIM-40 environment should adhere closely to this specification. Simplifying the motherboard will almost certainly compromise the operation of development systems, operating systems or applications software.

5.3 Basic Recommendations

There are many subsystems that the motherboard must provide a module with. ranging from simple functions (like power supplies, clocks etc.) through to host interfaces and JTAG services. This section details some recommendations for these.

5.3.1. Provision of Clocks

It is a requirement that a motherboard should be capable of supplying at least one clock frequency to the CLKIN pin of the supported modules, whether the motherboard supports shared memory or not. In either case, a single clock source will be adequate where that clock source can be altered - for example, if a socketed DIL oscillator module is used. Standard motherboards should provide the lowest common denominator frequency for this oscillator module.

This approach will allow any mix of modules to be successfully clocked from a single source. Where higher specification modules (e.g. using 50MHz C40s) axe available, the user has still got the option of replacing the clock module with one of a higher frequency.

More sophisticated motherboards could provide a range of clocks, which could be jumper selected by individual modules. This would ensure the maximum performance from the system with any module mix.

Them is no requirement in this specification for clocks of slave motherboards to be synchronous or to have low skew with that of the motherboard. This would be useful only where a shared memory system distributed across multiple motherboards is considered - this is not an area addressed by this specification.

5.3.2. Power Supplies

It is essential that all motherboards provide power to all the power supply pins implemented on the top and bottom connectors. All ground lines must be connected. The functionality of modules cannot be guaranteed where these lines are not implemented.

These power supplies must be adequately decoupled, and of low impedance to ensure correct operation of the module. Particular attention should be paid to the 12V supplies, as these are likely to be used to power any analogue circuitry on the module, and noise on these lines will directly affect the quality of this analogue interface. All power rails must be within 5% of the specified voltage to ensure compatibility with common semiconductor devices.

It may be that some host systems do not implement 12V rails. In these cases, the supply may be generated using DC_DC converters to amplify (and invert) the 5V rail. However, care must be exercised when using this technique to adequately smooth the outputs of the voltage converters.

5.3.3. Optional Global Memory Connector

It is left up to the individual hardware designer to decide whether or not to implement some form of global memory on his motherboard. Note that the signals on the global bus connector are unbuffered, taken direct from the processor. Implementing shared memory will require some form of bus arbitration, and possibly also require the signals to be buffered. Modules with both memory interfaces populated may suffer a performance penalty when used in this mode;however, the module mounted memory on the global port may itself become accessible to other devices.

This specification does not address memory shared between motherboards. While this may be feasible, it is felt to be well outside the scope of this document. Its implementation will depend largely on the format of the motherboard - for example, a VME format board would use that buses standard protocols.

5.3.4. Routing of communications ports on motherboards

Single width modules carrying a single C40 implement six bi-directional links. Given this number, it is relatively easy for the motherboard to implement full interconnect between its own module sites, leave links available for multiboard expansion, and still not use all the links on each module site.

In this case, it is suggested that as many module sites as possible on a motherboard implement communications port 0 as a link back to the ‘root’ device on the board; for example, the first site might communicate with a PC, so it should use link 0 for this; the second site communicates with the first, again through port 0; and so on.

It is also suggested that communications port 3 be given as free a path back to the root C40 as possible. This would allow a second port to be used by peripherals, and is particularly important for those slots which do not have link 0 access to the root processor.

If this scheme is implemented, any slot of a motherboard may be used by a module that implements only one link, allowing peripherals to be mounted on any motherboard site. Obviously, if some sort of crossbar switch is used with the links, this problem disappears.

5.3.5. Communications Port Termination on Motherboards

With a high speed communications port like that of the C40, care must be taken with its routing in the unbuffered state. It is essential that all motherboards implement pull-up resistors on the communications port control lines in order to ensure correct operation. This is the responsibility of the motherboard. Where a link is taken off-board, it is only necessary to provide pull-ups on those which default to inputs at RESET.

5.3.6. Bootstrap control

The bootstrap mode of the C40 device selects the source used for bootstrap code fetches. This is performed by the device reading the four IIOF lines shortly after RESET. Normally, a network will bootstrap from the communications ports; this is done by pulling all IIOF lines high. However, other bootstrapping modes are available, allowing the device to bootstrap from differing widths of ROM at various locations. To cater for these, the module must arrange for the appropriate settings on the IIOF lines. This is the module’s responsibility.

As a number of the IIOF lines are brought down to the motherboard, they must be held in high impedance after RESET. It is suggested that a flip flop controlled by the RESET* and IACK* lines could be used to enable/disable buffers to maintain high impedance during the bootstrap process.

5.4. Host Interfaces

5.4.1. General Description of interfaces.

The main interface on the TIM Motherboard is the one that joins the HOST machine’s bus to the first TMS320C40 in the system, thus forming a path for both data and control between the host and the TMS320C40 system.

There are two possible types of interface in such a system, a simple connection to the TMS320C40 link, and a shared memory type interface that occupies a larger address range of the host machine to allow ’Block Move’ type operations. The former would be I/O mapped in the PC environment, while the latter would always be memory mapped. The second interface may either be interfaced to a link or to an area of shared memory, possibly utilising the shared memory connector as defined in the TIM-40 spec.

The next major interface is to the system Reset structure. The motherboard reset scheme is defined so that all processors can be reset together to enable simple boot modes. This basic reset interface may be supplemented by a more sophisticated scheme, operating through the JTAG interface described later. This scheme could allow any processor from an arbitrarily large network to be reset at will, a feature that enables the user to build fault tolerant networks.

The third interface with the Host is the JTAG port. This allows the JTAG ports of the TMS320C40s to be exercised by various

software tools, including HLL debuggers and an individual reset driver. A standard connector for JTAG is defined to allow JTAG signals to be driven into or out of an individual motherboard.

For motherboards which provide the TMS320C40 links for external connection, a connector has been defined to carry the link.

The motherboard interfaces will generally be handled through device drivers, making the hardware details transparent to user software. This allows software compatibility while still using any special features built into specific motherboards. This section includes some example interfaces which can be regarded as a starting point, and which will provide the functionality required by such a driver.

5.4.2. Example IO interface to host

The basic requirement of a host is the ability to read and write data from the TMS320C40 link, without loss or corruption. The byte wide nature of the link makes the simplest interface a byte wide one, suiting many different host machines. This may not necessarily be the most efficient interface.

It must be remembered that TMS320C40 comport communication is always in 32 bit words, so any software must always perform reads and writes in blocks of four bytes.

There could be a separate address location for the write (from host to link) data and the read (from link to host) data. Each will have a location that gives the status of the interface, i.e. ready to receive data from the host, or ready to transfer data to the host.

The details of the interface (including the base address for this register block) will have to be chosen to suit each individual machine, but the registers could be organised as in:\USERS, offset from the chosen base address:

It is mandatory that the base address of such an interface should be selectable to avoid clashes in memory mapping. In an AT the possible base addresses could include a minimum of 0x150, 0x200 and 0x300 in the IO space.

Offset	Purpose	R/W	Notes
0	Read Data	R	
1	Write Data	W	
2	Input Status	R	Bit 7=1 - Data read for read
3	Output Status	R	Bit 7=1 - Ready for write
4	Config Status Register	R	Allows state of CONFIG to be read
5	Restart	R/W	Allows host to issue a global reset

TABLE 8 - Example Host Interface

Note that this example defines a simple byte-wide register-mapped interface between the host and the first C40, with status bits which indicate to the host whether it may read or write data. It is left up to the individual designer to decide whether or not to add some form of FIFO, perhaps emulating the C40's own FIFO scheme, or perhaps implement some form of interrupt or DMA capability for the host. The C40 end of the interface obviously implements these functions within the device itself.

As the interface software is implemented using device drivers, it is possible to modify this interface at will. For example, it would be possible to use 32-bit registers to communicate with the link, allowing one cycle/transfer. This would require a very simple modification to the device driver to cater for it.

5.4.3. RESET Control

The Reset Structure is a very important part of the TM-40 Motherboard specification. The aims were to allow a simple method of booting the entire network whilst enabling fault tolerant systems to reset any individual processor without disrupting the rest of the network. This is achieved by a combination of a global reset system, in conjunction with the JTAG subsystem described elsewhere.

Features of this system are that software may reset any processor in the system, without prior knowledge of the system connectivity and physical positioning of each processor in terms of motherboard number and slot number. This allows an operating system to allocate a processor ID code to each processor in the network, then request that a specific processor ID code be reset.

5.4.3.1. Global Reset Scheme

Global reset of processors within the network is achieved by having a system-wide RESET signal. This may be accessed from a single point by the host system; by toggling a bit in one of the control registers in the host interface, the system RESET line is asserted. To remove RESET and start all processors synchronously, the same bit should be toggled again.

The system-wide RESET line should be buffered on each motherboard, allowing it to maintain clean signal levels in even the largest system.

5.4.3.2. Individual Reset Scheme

The individual reset scheme is implemented through the JTAG interface, and as such is purely software; no additional hardware is required to support it. It is anticipated that many systems will choose not to implement this feature.

JTAG allows the user to contro1 the operation of any processor in the network at the lowest level. This feature may be used to examine registers within a given processor, load programs or reset a device.

Thus, to implement individual reset as required, host system could go through the following steps:

1) Host loads the operating system into the network **2)** Operating system assigns a unique processor ID to every CPU in the network.

It then informs the host that the network is initialised. **3)** Host uses JTAG interface to load each individual processor's ID code from that processor's memory. This ID code is then entered in a table with the relevant JTAG address for that processor. **4)** System then starts running application programs. As part of this, some part of the operating system monitors the network for processor or link failures. When one is detected, it informs the host of the failed processor's ID code. **5)** When informed of a link/cpu failure, the host uses the ID code as an index into the table of JTAG addresses formed earlier. It may now use JTAG to perform a reset on that individual processor.

Obviously, this is an involved process that involves a high degree of co-operation between the host, the network operating system, and the JTAG driver software. It also imposes some degree of overhead in that something must monitor the network status for link or processor failures. The overhead may be such that it is quicker to reboot the entire system than to reset a single processor. However, it may be that the advantages of being able to reset a single processor from a large group outweigh these disadvantages.

5.4.4. JTAG Interfacing

JTAG is the key to the C40 module's debug environment, and to the individual reset scheme described earlier. It is recommended that all motherboards provide a minimum of a header allowing access to the motherboard's JTAG system. A more sophisticated system would add some form of JTAG interface device, such as the 74ACT8990 Test Bus Controller (TBC), to the basic header.

As JTAG is a daisy chain, it is important that the chain is completed - even when a motherboard is not fully populated with modules. Some mechanism for bridging empty module sites must be implemented. This can be implemented using a jumper, allowing the JTAG path to be completed manually. However, the SENSE lines implemented on the module allow for automatic configuration of the JTAG path, by allowing the motherboard to disable or enable a buffer depending on whether a module is present or absent.

5.4.4.1. Simple JTAG Interface

As discussed earlier, it is essential for all motherboards to implement a minimum of a JTAG header, as shown in Figure 8. This would allow access to the modules on the motherboard via an emulator, like the XDS510. However, this does not allow the motherboard's host direct access to JTAG unless such an emulator is present.

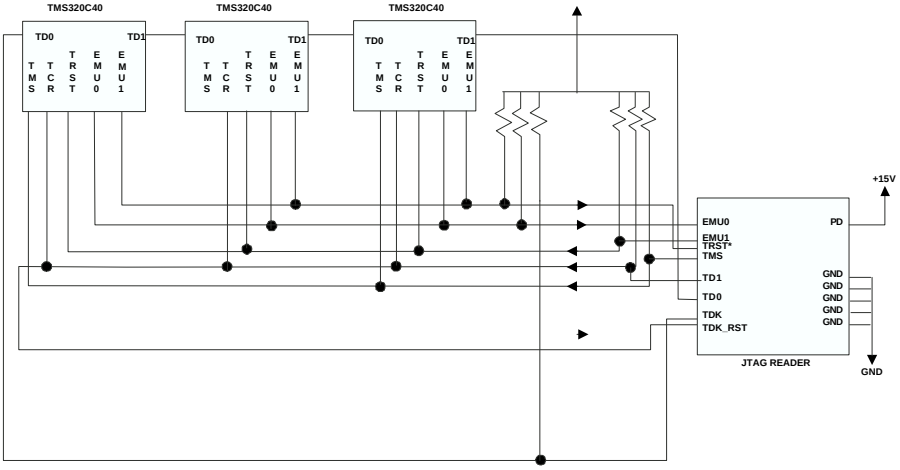

FIGURE 6 - Simple JTAG Interface

5.4.4.2. Master/Slave JTAG Debug Interface

Another possible JTAG configuration, again with multiple C40s in the JTAG path, would use the Test Bus Controller to control the JTAG chain. This could be interfaced to the C40 JTAG chain directly, but this would limit the expandability of the system; a better approach would be to implement an output header for the TBC signals, matching the JTAG Header in Figure 6. This design would allow the board to be used in three modes:

Stand-alone, with the TBC providing JTAG services for the modules on this board. This would be implemented using a jumper cable between the two headers.

JTAG Slave, where an external emulator or other JTAG master may be connected to the motherboard, allowing another JTAG control device access to this board's JTAG path;

JTAG Master, where the JTAG signals can be tapped off from this board, allowing another board to be included within the scan path controlled by this TBC.

For some machines or bus standards, the schematics for the interface between the TBC and the host may be available from Texas Instruments. Contact your sales representative for more details.

5.5. Multiple Motherboards

In larger systems, it becomes essential for more than one motherboard to be used. To provide a coherent debug strategy, and a consistent software interface to the system across multiple boards, the following guidelines should be adhered to.

5.5.1. Buffering Communications Ports

In order to ensure the correct operation of links across multiple boards or even racks containing multiple boards, it is advisable that communications ports be buffered. This involves decoding the state machine of the C40 to determine the direction of a link.

The buffered link would normally run in the standard C40 format, allowing buffered and unbuffered links to communicate directly. However, it may be desirable to transmit link data over considerable distances, in which case another format (e.g. fibre optic) may become desirable.

5.5.2. Connectors for Multiple Motherboards - Communications

Within a single rack, it is likely that ribbon cables will be used to connect motherboards together. However, between racks, more sophisticated interfaces (e.g. optical) may become common. In view of this, a standard connector has been specified for TIL-level links. This is a 14 pin dual row pin header, suitable for using Ribbon cable and IDC type connectors. See Table 9 for more details.

SIGNAL	PIN	PIN	SIGNAL
Data 0	1	2	Data 1
Data 2	3	4	Data 3
Data 4	5	6	Data 5
Data 6	7	8	Data 7
*CREQ	9	10	*CACK
*CSTRB	11	12	*CRDY
GND	13	14	GND

TABLE 9 – Connector Pinout

6. Hardware Compatibility Issues

Apart from the communications ports, several of the TMS320C40's on-chip resources are dedicated to system integration functions. This section details which resources are used, and what use is made of them.

6.1 Use of Interrupt Pins

6.1.1. NMI, IIOF0, IIOF1 & IIOF2

These interrupt/IO pins are taken out of the module on one of the connectors. They are available for use as required by the user's system. As IIOF lines axe used to control the bootstrap mode, it is essential that the motherboard should hold these in high impedance during the bootstrap procedure. This can be accomplished using flip-flops triggered by the module's RESET* and IACK* lines.

Additionally, it is necessary for the NM line to be pulled high through a resistor on the module. This will prevent the C40 receiving spurious interrupts in systems where NMI is unused on the motherboard.

6.1.2. IIOF3

This line is used by the module to drive the CONFIG line. It must be buffered using an inverting open-collector stage, to allow the line to be used as a motherboard-wide "configuring" signal. Once bootstrapped, external logic may be used to maintain the state of CONFIG, while IIOF3 could revert to an interrupt input.

6.2. Use of Timers

The timers am initialised by the ID ROM driver, from data stored in the ID ROM. There are specific memory locations reserved in the ROM for initialisation of the timer registers – see the appropriate section for more details.

6.2.1. Timer 0

Timer 0 is reserved for generating interrupts for software running on the module. It is intended for use as an operating system's context switching interrupt. It should be noted that the level of the pin TCLK0 after RESET is also important for the ID ROM driver. See section 4 for more details of this.

In view of the use of TCLK0 in decoding the ID ROM hardware, it is essential that when it is used for generating software system clocks the timer is initialised in such a way that TCLK0 can never carry the clock signal. This is easily achieved through the Timer Global Control Register. The timer must not be allowed to be driven extemally, as this may cause erroneous operation of the ID ROM.

A final point on TCLK0 is that the C40 actively drives this signal during ID EEPROM recall. It is therefore essential that the C40 be the only active driver on this line. In order to allow the line level to be set, it is recommended that TCLK0 be pulled high or low with a large value resistor. See section 4 for more details of the levels required on TCLK0.

6.2.2. Timer 1

Timer 1 is reserved for use by on-module hardware. An example of this might be generating a refresh counter for DRAM. If DRAM is not used, the clock may be used for other functions - for example, generating a sample rate for ADCs. This clock should be configured to be accessible from outside the TMS320C40. Typically, when used to generate a distributed refresh clock, this timer should typically be set up with a cycle time of 14us.

6.3 Memory in the Global Memory Environment

If a module does not use the global memory connector, the whole of the global memory port may be used for on-module memory. However, if a module does support the global connector, the module may still have its own bank of memory attached to the global port of the processor, but this must be accessed using STRB0*. It may be desirable to implement a mechanism to disable this bank of memory. This would allow total compatibility with motherboards which use STRB0* for memory expansion.

Generally, motherboards which support global memory expansion should use STRB1* for all accesses so as not to interfere with module memory.

It is highly desirable and recommended that the address and data lines are buffered on the motherboard, as the bus loading in a multiple module system may exceed the 80pF load the C40 is tested to. In addition, this allows C40s in a multi-module system to perform independent and concurrent accesses to their own on-module memory. However, the signals brought out to the global memory connector allow for an inexpensive non-buffered shared memory architecture. Attention should be paid to capacitive loading in this case.

7. Software Compatibility Issues

There are several issues which must be addressed to ensure low level software compatibility between modules. These are all concerned with the memory map, including where peripherals are placed in the map.

7.1 Memory Issues

To allow software to locate memory easily, standard guidelines are given below. It is recommended that main memory (RAM) should be built in a single contiguous pool on each bus, growing from fast memory (SRAM or on chip memory) at low addresses, through to the slowest memory (DRAM) at high addresses. This allows a smooth transition between fast memory, which tends to be scarce, and slower memory which is available in larger amounts.

7.1.1. Global Bus

Main memory (typically DRAM) located on the global bus should start at address 0x8000000, and build contiguously from there. Any faster memory (e.g. SRAM) should be located at 0x80000000, while slower memory should be placed above this. This allows the memory search program to establish easily where memory exists, and the size of that memory. Note that the C40 can directly address two banks of memory on this bus, using the on-chip strobes, chip selects, and page fault detection circuits.

Note also that the Global bus may be used in a shared memory system to access memory on the motherboard or indeed on other modules. In view of this, it is recommended that modules which implement the optional shared memory bus decode their memory below 0xC0000000.

7.1.2. Local Bus

Main memory on the local bus should again build upwards from the on-chip memories contiguously. This type of memory map allows a smooth expansion through memory, at the expense of increasing memory access times in higher memory.

Either strobe or page fault circuit may be used on the local bus.

7.2. Fast Memory Location

If one bank of fast memory (typically SRAM) is present, it is suggested that this be placed on the local bus, at address 0x00300000. This location is immediately after the internal memory, so fast memory will extend seamlessly from on-chip, through to off-chip expansion memory. Note that the on-chip memory is capable of multiple accesses per instruction cycle, while the off chip memory is not.

7.3. Bootstrap Memory Location

Non volatile memory is provided on every module in the form of the ID ROM. However, some modules will use the ROM bootstrap option of the TMS320C40. If this option is used, the ROM should be placed on any one of the standard bootstrap locations upwards.

The only restriction on this is that the bootstrap location selected should not conflict with the processor's memory pools, and that it should allow for any possible expansion of the module.

A point worth noting here is that after RESET, STRB1 and LSTRB 1 are disabled. Therefore, in order for the boot EPROM to be accessible at boot time, it must be located on either STRB0 or LSTRB0. However, it is desirable to place the EPROM in high memory to be clear of the memory pools, and for the EPROM to be accessible at the same time as the other memories installed on the module. This can be addressed using a logical OR of STRB0 and STRB1 or LSTRB0 and LSTRB1 to generate a single bus strobe, and using this to select the ROM in conjunction with memory decode; care must be taken to ensure that no other device can be selected at the same time!

7.4. Peripheral Location

It is recommended that peripherals be located on the local bus, in the block from 0x70000000 to 0x7fffffff, or on the global bus, from 0xB0000000 to 0xBFFFFFFF. These blocks will then be ignored by standard software memory search routines, preventing any memory test pattern writes which might cause invalid operation of peripherals. The areas are large enough to allow peripherals with large memory requirements, like multiple direct mapped frame buffers to be used.

7.5. Waitstate Generation

Unless otherwise specified by the data held in the ID ROM (described elsewhere), the bootstrap code will initialise the waitstate controllers for waitstates generated by software, giving the slowest bus cycle the C40 is capable of, but with that bus cycle terminating early should the addressed memory generate a READY signal.

APPENDIX A - CONNECTOR PINOUT

A.1 Notes to the Pinout

In order to reduce the number of pins required for the global memory connector, a number of compromises have had to be made. These are detailed below.

A.2 AE, DE*, CE0**

This are global bus enables direct from the C40. They should have a pulldown on the module to allow accesses to global memory (if present on the module; if no memory is present they could be pulled high to save power).

A3 CE10

This is the global bus enable for the second set of global strobes, direct from the C40. The motherboard should pull this line low to allow the module to access global memory. A high value pullup on the module can be used to lower power consumption.

A.4 RDY0 & RDY1**

These are direct from the C40. They should have a pullup on the module, unless the module requires the use of RDY0* to insert external wait states. In this case it is recommended that the use of RDY0~ by the module may be disabled to cater for the case where the motherboard requires it.

A.5 Driving AE, DE*, CE0*, CE1*, RDY0+ 4 RDY1**

The motherboard should either connect these signals direct to a logic Level (ground/Vcc), or use an active driver to control them. The use of open collector outputs will not guarantee proper system operation, as some lines may be pulled down by the module.

UDPI	1	J1	USER DEFINED PIN 01
VCC	2	J1	+5V POWER
UDP2	3	J1	USER DEFINED PIN 02
UDP3	4	J1	USER DEFINED PIN 03
UDP4	5	J1	USER DEFINED PIN 04
GND	6	J1	+0V POWER
UDP5	7	J1	USER DEFINED PIN 05
VCC	8	J1	+5V POWER
UDP6	9	J1	USER DEFINED PIN 06
VCC	10	J1	+5V POWER
GND	11	J1	+0V POWER
GND	12	J1	+0V POWER
GND	13	J1	+0V POWER
GND	14	J1	+0V POWER
TCK	15	J1	JTAG TEST CLOCK
VCC	16	J1	+5V POWER
EMU0	17	J1	EMULATION CONTROL 0
TMS	18	J1	JTAG MODE CONTROL
SENSE*	19	J1	PRESENSE LINE
GND	20	J1	+0V POWER
SENSE	21	J1	PRESENSE LINE
TDO	22	J1	TAG TEST OUTPUT
EMU1	23	J1	EMULATION CONTROL 1
VCC	24	J1	+5V POWER
TRST*	25	J1	JTAG RESET
TDI	26	J1	JTAG TEST INPUT
IACK*	72	J1	INTERRUPT ACKNOWLEDGE
GND	28	J1	+0V POWER
H3	29	J1	CLOCK OUTPUT H3
RESET8	30	J1	RESET INPUT
TCLK0	31	J1	TIMER 0 PIN
VCC	32	J1	+5V POWER
NM1*	33	J1	INTERRUPT - NON MUTABLE
TCLK1	34	J1	TIMER 1 PIN
CSTRB0*	35	J1	COMM PORT 0 STROBE
GND	36	J1	+0V POWER
CACK0*	37	J1	COMM PORT 0 ACKNOWLEDGE
CRDY0*	38	J1	COMM PORT 0 READY
CLKIN	39	J1	CLOCK INPUT

VCC	40	J1	+5V POWER
C0D7	41	J1	COMM PORT 0,DATA BIT 7
CREQ0*	42	J1	COMM PORT 0 REQUEST
C0D5	43	J1	COMM PORT 0, DATA BIT 5
GND	44	J1	+0V POWER
C0D1	45	J1	COMM PORT 0, DATA BIT 1
C0D6	46	J1	COMM PORT 0, DATA BIT 6
C0D3	47	J1	COMM PORT 0, DATA BIT 3
VCC	48	J1	+5V POWER
C0D2	49	J1	COMM PORT 0, DATA BIT 2
C0D4	50	J1	COMM PORT 0, DATA BIT 4
C0D0	51	J1	COMM PORT 0, DATA BIT 0
GND	52	J1	+0V POWER
H1	53	J1	CLOCK OUTPUT H1
IIOF1	54	J1	INTERRUPT I/O FLAG 1
IIOF0	55	J1	INTERRUPT I/O FLAG 0
VCC	56	J1	+5V POWER
CRDY3*	57	J1	COMM PORT 3 READY
CACK3*	58	J1	COMM PORT 3 ACKNOWLEDGE
CREQ3*	59	J1	COMM PORT 3 REQUEST
GND	60	J1	+0V POWER
C3D0	61	J1	COMM PORT 3, DATA BIT 0
CSTRB3*	62	J1	COMM PORT 3 STROBE
C3D2	63	J1	COMM PORT 3, DATA BIT 2
VCC	64	J1	+5V POWER
C3D4	65	J1	COMM PORT 3, DATA BIT 4
C3D1	66	J1	COMM PORT 3, DATA BIT 1
C3D3	67	J1	COMM PORT3, DATA BIT 3
GND	68	J1	+0V POWER
C3D5	69	J1	COMM PORT 3, DATA BIT 5
C3D6	70	J1	COMM PORT 3, DATA BIT 6
C3D7	71	J1	COMM PORT 3, DATA BIT 7
VCC	72	J1	+5V POWER
IIOF2	73	J1	INTERRUPT I/O FLAG 2
CONFIG*	74	J1	SYSTEM CONFIG LINE
GND	75	J1	+0V POWER
GND	76	J1	+0V POWER
-12V	77	J1	-12V POWER LINE
+12V	78	J1	+12V POWER LINE

-12V	79	J1	-12V POWER LINE

+12V	80	J1	+12V POWER LINE

UDP7	1	J2	USER DEFINED PIN 07
UDP8	2	J2	USER DEFINED PIN 08
UDP9	3	J2	USER DEFINED PIN 09
UDP10	4	J2	USER DEFINED PIN 10
GND	5	J2	+0V POWER
UDP11	6	J2	USER DEFINED PIN 11
VCC	7	J2	+5V POWER
UDP12	8	J2	USER DEINED PIN 12
VCC	9	J2	+5V POWER
GND	10	J2	+0V POWER
C5D0	11	J2	COMM PORT 5, DATA BIT 0
GND	12	J2	+0V POWER
GND	13	J2	+OV POWER
C5D2	14	J2	COMM PORT 5, DATA BIT 2
CRDY5*	15	J2	COMM PORT 5 READY
C5D4	16	J2	COMM PORT 5, DATA BIT 4
VCC	17	J2	+5V POWER
CACK5*	18	J2	COMM PORT 5 ACKNOWLEDGE
CREQ5*	19	J2	COMM PORT 5 REQUEST
C5D7	20	J2	COMM PORT 5, DATA BIT 7
GND	21	J2	+0V POWER
CSTRB5*	22	J2	COMM PORT 5 STROBE
C5D5	23	J2	COMM PORT 5, DATA BIT 5
C5D6	24	J2	COMM PORT 5, DATA BIT 6
VCC	25	J2	+5V POWER
C5D3	26	J2	COMM PORT 5, DATA BIT 3
C4D7	27	J2	COMM PORT 4, DATA BIT 7
C5D1	28	J2	COMM PORT 5, DATA BIT 1
GND	29	J2	+0V POWER
C4D3	30	J2	COMM PORT 4, DATA BIT 3
CACK4*	31	J2	COMM PORT 4 ACKNOWLEDGE
CRDY4*	32	J2	COMM PORT 4 READY
VCC	33	J2	+5V POWER
C4D6	34	J2	COMM PORT 4, DATA BIT 6
C4D4	35	J2	COMM PORT 4, DATA BIT 4
CREQ4*	36	J2	COMM PORT 4 REQUEST
GND	37	J2	+0V POWER
CSTRB4*	38	J2	COMM PORT 4 STROBE
D4D1	39	J2	COMM PORT 4, DATA BIT 1
C4D0	40	J2	COMM PORT 4, DATA BIT 0
VCC	41	J2	+5V POWER

C4D5	42	J2	COMM PORT 4, DATA BIT 5
C2D6	43	J2	COMM PORT 2, DATA BIT 6
C4D2	44	J2	COMM PORT 4, DATA BIT 2
GND	45	J2	+0V POWER
C2D2	46	J2	COMM PORT 2, DATA BIT 2
C2D3	47	J2	COMM PORT 2, DATA BIT 3
C2D4	48	J2	COMM PORT 2, DATA BIT 4
VCC	49	J2	+5V POWER
C2D5	50	J2	COMM POTR 5, DATA BIT 5
C2D0	51	J2	COMM PORT 2, DATA BIT 0
C2D7	52	J2	COMM POTR 2, DATA BIT 7
GND	53	J2	+0V POWER
C2D1	54	J2	COMM PORT 2, DATA BIT 1
CREQ2*	55	J2	COMM PORT 2 REQUEST
CRDY2*	56	J2	COMM PORT 2 READY
VCC	57	J2	+5V POWER
CSTRB2*	58	J2	COMM PORT 2 STROBE
CSTRB1*	59	J2	COMM PORT 1 STROBE
CACK2*	60	J2	COMM PORT 2 ACKNOWLEDGE
GND	61	J2	+0V POWER
CRDY1*	62	J2	COMM PORT 1 READY
C1D3	63	J2	COMM PORT 1, DATA BIT 3
CACK1*	64	J2	COMM PORT 1 ACKNOWLEDGE
VCC	65	J2	+5V POWER
CREQ1*	66	J2	COMM PORT 1 REQUEST
C1D5	67	J2	COMM PORT 1, DATA BIT 5
C1D7	68	J2	COMM PORT 1, DATA BIT 7
GND	69	J2	+OV POWER
C1D4	70	J2	COMM PORT 1, DATA BIT 4
C1D6	71	J2	COMM PORT 1, DATA BIT 6
C1D1	72	J2	COMM PORT 1, DATA BIT 1
VCC	73	J2	+5V POWER
C1D2	74	J2	COMM PORT 1, DATA BIT 2
GND	75	J2	+0V POWER
C1D0	76	J2	COMM PORT 1, DATA BIT 0
GND	77	J2	+0V POWER
GND	78	J2	+0V POWER
GND	79	J2	+0V POWER
GND	80	J2	+0V POWER

A30	1	J3	GLOBAL ADDRESS BUS-BIT 30
A29	2	J3	GLOBAL ADDRESS BUS- BIT 29
A28	3	J3	GLOBAL ADDRESS BUS-BIT 28
A27	4	J3	GLOBAL ADDRESS BUS-BIT 27
A26	5	J3	GLOBAL ADDRESS BUS-BIT 26
A25	6	J3	GLOBAL ADDRESS BUS-BIT 25
A24	7	J3	GLOBAL ADDRESS BUS-BIT 24
A23	8	J3	GLOBAL ADDRESS BUS-BIT 23
A22	9	J3	GLOBAL ADDRESS BUS-BIT 22
A21	10	J3	GLOBAL ADDRESS BUS-BIT 21
A20	11	J3	GLOBAL ADDRESS BUS-BIT 20
A19	12	J3	GLOBAL ADDRESS BUS-BIT 19
A18	13	J3	GLOBAL ADDRESS BUS-BIT 18
A17	14	J3	GLOBAL ADDRESS BUS-BIT 17
A16	15	J3	GLOBAL ADDRESS BUS-BIT 16
A15	16	J3	GLOBAL ADDRESS BUS-BIT 15
A14	17	J3	GLOBAL ADDRESS BUS-BIT 14
A13	18	J3	GLOBAL ADDRESS BUS-BIT 13
A12	19	J3	GLOBAL ADDRESS BUS-BIT 12
A11	20	J3	GLOBAL ADDRESS BUS-BIT 11
A10	21	J3	GLOBAL ADDRESS BUS-BIT 10
A09	22	J3	GLOBAL ADDRESS BUS-BIT 09
A08	23	J3	GLOBAL ADDRESS BUS-BIT 08
A07	24	J3	GLOBAL ADDRESS BUS-BIT 07
A06	25	J3	GLOBAL ADDRESS BUS-BIT 06
A05	26	J3	GLOBAL ADDRESS BUS-BIT 05
A04	27	J3	GLOBAL ADDRESS BUS-BIT 04
A03	28	J3	GLOBAL ADDRESS BUS-BIT 03
A02	29	J3	GLOBAL ADDRESS BUS-BIT 02
A01	30	J3	GLOBAL ADDRESS BUS-BIT 01
A00	31	J3	GLOBAL ADDRESS BUS-BIT 00
D31	32	J3	GLOBAL DATA BUS-BIT 31
D30	33	J3	GLOBAL DATA BUS-BIT 30
D29	34	J3	GLOBAL DATA BUS-BIT 29
D28	35	J3	GLOBAL DATA BUS-BIT 28
D27	36	J3	GLOBAL DATA BUS-BIT 27
D26	37	J3	GLOBAL DATA BUS-BIT 26
D25	38	J3	GLOBAL DATA BUS-BIT 25
D24	39	J3	GLOBAL DATA BUS-BIT 24
D23	40	J3	GLOBAL DATA BUS-BIT 23
D22	41	J3	GLOBAL DATA BUS-BIT 22

D21	42	J3	GLOBAL DATA BUS-BIT 21
D20	43	J3	GLOBAL DATA BUS-BIT 20
D19	44	J3	GLOBAL DATA BUS-BIT 19
D18	45	J3	GLOBAL DATA BUS-BIT 18
D17	46	J3	GLOBAL DATA BUS-BIT 17
D16	47	J3	GLOBAL DATA BUS-BIT 16
D15	48	J3	GLOBAL DATA BUS-BIT 15
D14	49	J3	GLOBAL DATA BUS-BIT 14
D13	50	J3	GLOBAL DATA BUS-BIT 13
D12	51	J3	GLOBAL DATA BUS-BIT 12
D11	52	J3	GLOBAL DATA BUS-BIT 11
D10	53	J3	GLOBAL DATA BUS-BIT 10
D09	54	J3	GLOBAL DATA BUS-BIT 09
D08	55	J3	GLOBAL DATA BUS-BIT 08
D07	56	J3	GLOBAL DATA BUS-BIT 07
D06	57	J3	GLOBAL DATA BUS-BIT 06
D05	58	J3	GLOBAL DATA BUS-BIT 05
D04	59	J3	GLOBAL DATA BUS-BIT 04
D03	60	J3	GLOBAL DATA BUS-BIT 03
D02	61	J3	GLOBAL DATA BUS-BIT 02
D01	62	J3	GLOBAL DATA BUS-BIT 01
D00	63	J3	GLOBAL DATA BUS-BIT 00
CE1*	64	J3	GLOBAL BUS - CONTROL ENABLE 1
LOCK*	65	J3	GLOBAL BUS - LOCK
RDY1*	66	J3	GLOBAL BUS - READY
RDY0*	67	J3	GLOBAL BUS - READY 0
CE0*	68	J3	GLOBAL BUS - CONTROL ENABLE 0
DE*	69	J3	GLOBAL BUS - DATA BUS ENABLE
PAGE1	70	J3	GLOBAL BUS - PAGE ENABLE 1
STRB1*	71	J3	GLOBAL BUS - STROBE 1
RNW1	72	J3	GLOBAL BUS - READ/WRITE* 1
STRB0*	73	J3	GLOBAL BUS - STROBE 0
STAT0	74	J3	GLOBAL BUS - SATAUS LINE 0
STAT1	75	J3	GLOBAL BUS - SATAUS LINE 1
STAT2	76	J3	GLOBAL BUS - SATAUS LINE 2
STAT3	77	J3	GLOBAL BUS - SATAUS LINE 3
RNW0	78	J3	GLOBAL BUS - READ/WRITE* 0
PAGE0	79	J3	GLOBAL BUS - PAGE ENABLE 1
AE*	80	J3	GLOBAL ADDRESS BUS ENABLE

8. Addendum to the TIM-40

About this Addendum

This document should be used as an addendum to the TIM-40 TMS320 Module Specification version 1.01. It contains specifics on implementation of TIM-40 modules based on the TMX320C44 processor. The TMX320C44 processor differs from the TMS320C40 in the following features relevant to TIM-40 module design:

- 4 communication ports (C44) vs 6 communication ports (C40)
- Different on-chip Boot loader mode selection
- A port direction pin (CDIRx), one for each communication port. These pins are not present in C40 devices.
- C44 external address buses are 24-bit wide vs the 31-bit wide external address buses of the C40.

If you have questions or comments about this document or if you wish to register as a new TIM-40 manufacturer, write to:

TIM-40 Co-ordinator 12203 SW Freeway MS 737 Stafford, TX 77001

or fax your request to: **(713)274-2279 Attention: TIM-40 Co-ordinator**

Addendum Content

This document contains recommendations regarding the following items:

Item 0: C44 TIM-40 naming convention

Item 1: C44 TM-40 ID-ROM address

Item 2: C44 TIM-40 Processor ID

Item 3: C44 TIM-40 recommendation on TIM-40 module A24-A30 signals

Item 4: C44 TIM-40 commport numbering (uniprocessor modules)

Item 5: C44 TIM-40 commport numbering (dual C44 single-width modules)

Item 6: C44 CDIRx pin-out allocation

Item7:C44bootloadermodeselection

Items 0, 1, 2 and 7 must be adhered to by all TIM-40 manufacturers. Items 3, 4, 5 and 6 are presented as *recommendations* not as rules that the TIM-40 manufacturer must enforce. This decision was taken to allow existing C40-targeted motheboards to hold C44 TIM-40 modules. However, TI encourages existing and emerging TIM-40 manufacturers to follow these *recommendations* to keep as much compatibility as possible among different third party products.

8.1 Naming Convention

The *TIM-40 standard* name should be maintained even when referring to modules carrying a C44 product. A C44-based module should be referred to as either *C44 TIM-40 module* or *TMS320C44 TIM-40 module..*

8.2 ID-ROM Address

The ID-ROM address will continue being Ox7000 0000. A C44 access to this address causes an external access at address OxO. Section 4.9.3 of the TIM-40 spec v1.01 recommends the use of the defaults Local Bus Control Register to access the ID-ROM. Therefore LSTRBO should be used to access the ID-ROM.

8.3 Process or ID.

The C44 processor ID code is 6.

8.4 A24-A30 signals

J3 pins 1,2,3,4,5,6,7 can be driven by an extra circuitry on the module. This extra circuitry provides the ability to access the full address range of off-module global resources (shared-memory) for existing TIM-40 motherboards. This adheres to the original TIM-40 spec recommendation of placement of the shared-memory at address >= OxCOOO 0000.

Implementation of this external circuitry is manufacturer specific. TI suggests the use of a *page register* The end-user could write an specific value to a page register. This guarantees compatibility among different TIM- 40 vendors. The page register on the TIM-40 module could drive a valid value on lines A24 through A30. The page register could be written via the local or global bus. Be aware that:

- A24-A30 should not be driven unless the AE signal is driven low.
- If the page register is located in the local bus, there is a chance that a local access will change the page register value at the same time as an off-module global access which uses the register contents. Caution should be taken in this case to avoid unpredictable results.

8.5 Commport numbering (uniprocessor C44 TIM-40 module)

Due to the absence of commports 0 and 3 in the C44, commport renumbering is recommended. Commport renumbering is recommended for 2 reasons:

- If no renumbering is done, TIM-40 link 0 and 3 will be unconnected, contradicting the recommendation given in section 5.3.4 of the TIM-40 spec version 1.01. This could cause incompatibilities with existing motherboards.
- Dual C44 single-width TIM-40 modules are being introduced. These modules also require commport renumbering as covered in item 5)

TI proposes the following renumbering scheme *only* for C44 modules. This renumbering scheme restricts the renumbering to TIM-40 links 0 and 3 to minimise confusion among TIM-40 end-users regarding board commport connectivity

TIM - 40 Pinout	**Signal**
TIM - 40 link 0	C44 link 2
TIM - 40 link 1	C44 link 1
TIM - 40 link 2	TIM - 40 link 5 (*1)
TIM - 40 link 3	C44 link 5
TIM - 40 link 4	C44 link 4
TIM - 40 link 5	TIM - 40 link 2 (*1)

*1: Optional. If TIM-40 link 5 and 11M-40 link 2 are not connected, then pull-ups should be placed in C, CACK, CSTRB and CRDY of both commports.

8.6 Commport numbering (dual C44 single-width TIM-40 module)

The introduction of new dual C44 single-width TIM-40 modules forces the renumbering of the TIM-40 link pinout. Following is TI recommendation regarding dual C44 single-width TIM-40 modules. The 2 C44s are referred to as C44-A and C44-B:

TIM - 40 pinout	**Signal**
TIM - 40 link 0	C44 - B link 1
TIM - 40 link 1	C44 - B link 2
TIM - 40 link 2	C44 - A link 2
TIM - 40 link 3	C44 - A link 4
TIM - 40 link 4	C44 - A link 5
TIM - 40 link 5	C44 - B link 5

C44 - B link 4 connects to C44 - A link 1.

8.7 CDIR pinout allocation

A commport direction pin (CDIR1, CDIR2, CDIR4, CDIR5) is available for each C44 communication ports. These signals are output signals that indicate if a communication port is in output mode (transmitting) or in input mode (receiving). Having these signals off-module is essential in some cases because it eases commport interfacing and buffering.

TI recommends the assignment of the following UDPs (User Defined Pins) as follows:

UDP pin	**New assignment**
UDP 6	C44SENSE*pin
UDP 7	CDIR 1
UDP 8	CDIR 2
UDP 9	CDIR 4
UDP 10	CDIR 5
UDP 11	reserved (*2)
UDP 12	reserved (*2)

(*2) reserved for the future C4x spinoffs and/or TIM - 40 spec change

ln order to recognise that these UDP pins are valid CDIRx signals and not old floating pins, Tl advises the use of an UDP pin as a C44SENSE* pin. recommends the use of UDP 6 for this purpose. However. if this conflicts with a previous UDP 6 assignment. The election of another UDP pin is left to the discretion of the TIM-40 manufacturer. This C44SENSE* pin could be driven low by a C44 TIM - 40 module. New TIM-40 motherboards would pull the C44SENSE* pin high resistively, so if the signal is high, then a C40 TM-40 module is in place.

Item 7. C44 Bootloader Mode Selection

Following is the bootloader mode selection supported by C44 devices with device revisions higher than or equal to 1.1 Boot loader mode selection in C44 device revision 1.0 is the same as for the TMS320C40 as described in the C4x User's Guide. The new C44 bootloader mode selection enhances the boot loader selection choices while keeping the main bootloader mode choices of old revisions.

IIOF3	**IIFO2**	**IIFO1**	**IIFO0**	**Source program location**
1	1	0	1	30 0000h 1strb0
1	0	1	1	00 0000h 1strb0
1	0	0	1	80 0000h strb0
0	1	1	1	80000 0000h strb0
0	1	0	1	8040 0000h strb0
0	0	1	1	8080 0000h strb0
0	0	0	1	reserved
1	1	1	1	comm port