

**inmos**<sup>®</sup>

# **T9000 DEVELOPMENT TOOLS**

Preliminary Datasheets



INMOS is a member of the SGS-THOMSON Microelectronics Group

## **INMOS transputer databook series**

Transputer Databook

Military and Space Transputer Databook

Transputer Development and *iq* Systems Databook

Transputer Applications Notebook: Architecture and Software

Transputer Applications Notebook: Systems and Performance

T9000 Transputer Hardware Reference Manual


T9000 Transputer Instruction Set Manual

T9000 Development Tools – Preliminary Datasheets

INMOS reserves the right to make changes in specifications at any time and without notice. The information furnished by INMOS in this publication is believed to be accurate; however, no responsibility is assumed for its use, nor for any infringement of patents or other rights of third parties resulting from its use. No licence is granted under any patents, trademarks or other rights of INMOS.

INMOS Limited 1993

 , IMS, occam and DS-Link are trademarks of INMOS Limited.

 is a registered trademark of the SGS-THOMSON Microelectronics Group.

INMOS Limited is a member of the SGS-THOMSON Microelectronics Group.

INMOS document number: 72 TRN 249 00

**ORDER CODE:DBT9000DTST/1**

# Contents

<b>Preface</b> .....	<b>iii</b>
<b>1 Introduction</b> .....	<b>1</b>
1.1 Transputer modules .....	2
1.2 IMS T9000 board products .....	2
1.3 IMS T9000 software .....	3
<b>2 IMS Dx394 T9000 ANSI C Toolset</b> .....	<b>5</b>
2.1 Introduction .....	6
2.2 IMS T9000 networks .....	8
2.3 Parallel programming .....	9
2.4 Compiler and compilation tools .....	12
2.5 Libraries .....	16
2.6 Configuration, initialization and loading .....	16
2.7 Connecting IMS T9000 networks to T2/T4/T8 networks .....	21
2.8 Product components .....	22
2.9 Product variants .....	23
2.10 Problem reporting and field support .....	24
<b>3 IMS Dx395 T9000 occam 2 Toolset</b> .....	<b>25</b>
3.1 Introduction .....	26
3.2 IMS T9000 networks .....	28
3.3 Parallel programming .....	30
3.4 Compiler and compilation tools .....	31
3.5 Libraries .....	33
3.6 Configuration, initialization and loading .....	34
3.7 Connecting IMS T9000 networks to T2/T4/T8 networks .....	38
3.8 Product components .....	40
3.9 Product variants .....	40
3.10 Problem reporting and field support .....	42
<b>4 IMS D4390 T9000 INQUEST</b> .....	<b>43</b>
4.1 Product overview .....	44
4.2 Interactive windowing debugger .....	44
4.3 Performance analysis tools .....	48
4.4 Product details .....	49
4.5 Problem reporting and field support .....	49

---

<b>5</b>	<b>IMS B490 T9000 Development TRAM</b>	<b>51</b>
5.1	Introduction	52
5.2	IMS B490 operation	52
5.3	Support software	53
5.4	Hardware description	58
5.5	Pin descriptions	61
5.6	Mechanical details	62
5.7	Installation	64
5.8	Specification	65
5.9	Ordering Information	66
5.10	Field Support	66
5.11	References	66
<b>6</b>	<b>IMS B926 4Mbyte HTRAM</b>	<b>67</b>
6.1	Introduction	68
6.2	Specification	70
6.3	Ordering Information	74
6.4	Field Support	74
6.5	References	74
<b>7</b>	<b>IMS B100 VME Format Motherboard</b>	<b>75</b>
7.1	Introduction	76
7.2	Hardware Specification	78
7.3	Ordering Information	82
7.4	Field Support	82
7.5	References	83
<b>A</b>	<b>HTRAM specification</b>	<b>85</b>
A.1	Introduction	86
A.2	Interface Signals	86
A.3	Dimensions	89
A.4	Pinout	96
A.5	Power Supplies	101
A.6	Operating Environment	102
A.7	Initialization and Bootstrap	102
A.8	Specification of HTRAMs	106
A.9	References	106

---

## **Preface**

This booklet describes the INMOS board hardware and software tools provided for development of systems using IMS T9000 transputers. The products described are:–

### **Software products**

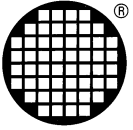
- IMS Dx394 T9000 ANSI C Toolset
- IMS Dx395 T9000 occam 2 Toolset
- IMS D4390 T9000 INQUEST development environment

### **Board products**

- IMS B490 T9000 Development TRAM
- IMS B926 4Mbyte HTRAM
- IMS B100 VME format HTRAM motherboard

The specification of the HTRAM format is included in an appendix.





# Introduction

This document gives advance information about the IMS T9000 development systems. These products are under continuing development and product details may change at any time without notice.

The new board and software products described in this booklet are suitable for constructing and programming IMS T9000 systems. An IMS T9000 system may use a single IMS T9000 or a network of IMS T9000 transputers.

## 1.1 Transputer modules

A module standard, the TRAM (TRANsputer Module), was launched with the transputer to offer systems developers a quick and easy way to build transputer development systems hardware. TRAMs are, in effect, board-level transputers with a simple, standardized interface. They integrate processor, memory and peripheral functions allowing powerful, flexible transputer-based systems to be produced with a minimum of design effort. One or more TRAMs may be plugged into slots on a TRAM motherboard, which may include an interface to a host. Thus single or multi-transputer systems may be built very quickly from a range of standard products.

## 1.2 IMS T9000 board products

The IMS T9000 offers a much enhanced communications architecture using the new links, called DS-Links (Data/Strobe). These links cannot be directly connected to IMS T2xx / T4xx / T8xx transputer networks, which use OS-Links (Over Sampled). In order to take full advantage of the new communications architecture, new board-level hardware and software development tools will be introduced which function alongside existing hardware and software. It is envisaged that IMS T9000 users will adopt one of the following options:

- Use only IMS T9000 family devices.
- Connect an IMS T2xx/T4xx/T8xx network to an IMS T9000 family network, using an IMS C100 system protocol converter as a gateway between the networks.
- Build an IMS T9000 network with IMS T2xx/T4xx/T8xx products as peripherals, using IMS C100 system protocol converters as interfaces.

INMOS is introducing a range of IMS T9000 products based on the existing TRAM standard to minimize the cost for existing transputer users. A new HTRAM (High performance TRANsputer Module) standard has been defined to obtain optimal performance from the IMS T9000 super-scalar processor and enhanced communications architecture. A range of HTRAM modules and motherboards is being introduced.

Several host / transputer interfaces are available based on the TRAM standard. New HTRAM format motherboards with host interfaces will also be introduced. In the meantime, the IMS B490 T9000 Development TRAM (described below) can be used within TRAM-based systems to give a host / IMS T9000 interface.

### 1.2.1 IMS B490 T9000 Development TRAM

The IMS B490 T9000 Development TRAM is the primary board-level product for IMS T9000 development systems. It provides quick and easy IMS T9000 access using any one of the range of existing TRAM-based architectures. The T9000 Development TRAM can be:

- used alone, for developing single IMS T9000 applications
- used to allow a host to control and communicate with an IMS T9000 network or
- used as an element of a TRAM-based network.

The T9000 Development TRAM will plug directly into existing TRAM motherboard slots. It may be used as a single processor or as an element of a TRAM network. In addition, the DS-Links (including the control



links) from the IMS T9000 transputer are brought out from IMS B490 T9000 Development TRAM using on-board hardware connectors. The IMS T9000 on the IMS B490 T9000 Development TRAM can be thus connected externally using DS-Link flying leads to other IMS T9000 transputers. A network of IMS T9000 transputers may be built by connecting together the DS-Links on several T9000 Development TRAMs

A single T9000 Development TRAM can be used as an interface to an HTRAM network or custom-built IMS T9000 hardware. Typically, a host / IMS T9000 interface would consist of a T9000 Development TRAM and a host / transputer interface which provides OS-Links, such as the IMS B008 PC TRAM Motherboard or IMS B300 Ethernet to Transputer Gateway. Together, these provide a host / IMS T9000 interface with separate control and data link connections.

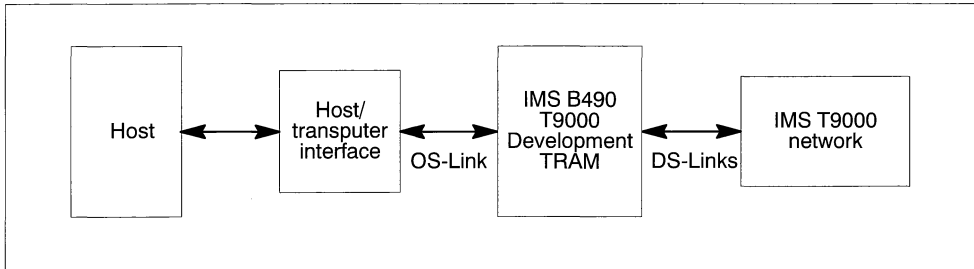


Figure 1.1 A host / IMS T9000 interface

The IMS B490 T9000 Development TRAM and accompanying software are fully described in Chapter 5.

## 1.2.2 HTRAMs

HTRAM (High performance TRANsputer Module) is the new transputer module standard for IMS T9000s, using the same principles as the TRAM. An HTRAM module uses only DS-Links and plugs into one or more slots in an HTRAM motherboard. These products use DS-Links for transmitting data and control packets through processor networks. This allows IMS T9000 development systems and networks to be easily constructed. Using HTRAMs allows the full power of IMS T9000 processing and communications performance to be obtained.

The HTRAM standard defines a mother / daughter board format which has the following features:

- Space-efficient modules
- Simple, easy-to-use interface
- Easy system integration
- Adopted by many companies world-wide

The IMS B926 4Mbyte HTRAM is described in Chapter 6. The IMS B100 VME format HTRAM Motherboard is described in Chapter 7. The HTRAM specification is included in appendix A.

## 1.3 IMS T9000 software

IMS T9000 systems developers may use the INMOS family of T9000 development software to program, debug and control application programs. Three main products are used to program IMS T9000 transputer devices: INMOS T9000 ANSI C and OCCam 2 Toolsets and T9000 INQUEST development environment. These products are source and usage compatible with software products for the IMS T2xx/T4xx/T8xx transputer family.

INMOS T9000 ANSI C and OCCam 2 Toolsets allow programs to be written, configured and run on one or more IMS T9000 transputers. Both these toolsets have been tuned to exploit the performance and flexibility offered by the IMS T9000 processor and communications architecture.

ANSI C is the standard language for developers of standard portable real-time multi-threading applications. OCCAM 2 is an efficient language with high-level facilities to support parallelism and communications.

The INMOS T9000 ANSI C and OCCAM 2 Toolsets are described in Chapters 2 and 3 respectively.

Development systems can be hosted by one of a range of supported hosts. The INMOS T9000 ANSI C and OCCAM 2 Toolset tools run on the host and the built code is loaded onto the target transputer system for testing. The final system may be hosted or support is given for building stand-alone systems which boot and load from ROM.

T9000 INQUEST is an advanced development environment featuring excellent debugging and profiling facilities. T9000 INQUEST enhances the standard ANSI C and occam 2 Toolsets, giving the user the latest in multi-process, multi-processor profiling and debugging tools with advanced windowing user interface operation.

The T9000 INQUEST development environment is described in Chapter 4.



## 2.1 Introduction

This document contains advance information for the INMOS T9000 ANSI C Toolset. Within this document the current IMS T2xx, T4xx and T8xx transputers are called 'T2/T4/T8' transputers.

The INMOS T9000 ANSI C Toolset provides a complete ANSI C cross-development system for IMS T9000 transputer network targets. It can be used to build sequential or parallel programs for single IMS T9000 transputers or for multi-transputer IMS T9000 networks.

The Toolset supports the construction of parallel C programs, which may be loaded onto IMS T9000 networks via a link or put into a form suitable for booting from ROM. The Toolset also includes the hardware configuration tools required to initialize IMS T9000 networks. It fully supports the use of IMS C104 packet routing switches including labelling and initializing. The Toolset also supports the use of the IMS C100 system protocol converter device which provides a means to connect T2/T4/T8 networks to IMS T9000 networks.

### 2.1.1 Key features

- Full ANSI C cross-compiler (X3.159-1989) for IMS T9000 target networks
- Validated against Plum Hall validation suite
- Code generation for IMS T9000 instruction set and pipelined CPU
- Excellent compile time diagnostics
- Global and local optimization
- Support for parallelism
- Assembler
- Support for assembler inserts
- Listings of where variables and functions reside in memory
- Small runtime overhead
- Separate compilation, using linker and librarian tools
- Tools for creating and loading multi-processor programs
- Exploitation of the new communications architecture by the configuration tools
- Software routing of channels for networks without routing chips
- Automatic makefile generator
- Mixed language programming support
- Tools to support preparation of programs for EPROM
- Consistent tools across PC and Sun-4 hosts
- Support for dynamically loading programs and functions

### 2.1.2 Overview

The IMS T9000 transputer range of devices will be supported by the following software products:

- IMS Dx394 T9000 ANSI C Toolset

- IMS Dx395 T9000 occam 2 Toolset
- IMS Dx390 T9000 INQUEST development environment, containing debugging and profiling tools

In addition there will be other software products supporting the loading, running and serving of programs over specific host / IMS T9000 interface boards. This software will be supplied with the hardware.

The INMOS T9000 ANSI C Toolset is source compatible with INMOS T2/T4/T8 Toolset products. This has been achieved by using similar compiler, libraries, linker, configurator and other tools. Code written for a single IMS T4xx or IMS T8xx can be ported directly to an IMS T9000 for performance enhancement.

The IMS T9000 offers two important enhancements; the new communications architecture and the new pipelined CPU. In order to take full advantage of these features, a new Toolset has been developed which will function alongside the existing Dx394 T9000 ANSI C Toolset. It is envisaged that IMS T9000 users will adopt one of the following options, which are all supported by the software:

- Use only IMS T9000 family devices. Users of T2/T4/T8 networks can recompile existing T2/T4/T8 code for a new IMS T9000 network.
- Connect a T2/T4/T8 network to an IMS T9000 family network, using an IMS C100 system protocol converter as a gateway between the networks.
- Build an IMS T9000 network with T2/T4/T8 products as peripherals, using IMS C100 system protocol converters as interfaces.

The configuration tools are the main area in which the difference between the IMS T9000 and the T2/T4/T8 range is apparent to the programmer. *Configuration* is the process of mapping a multi-process network application to a transputer network. Tools are provided in the INMOS T9000 ANSI C Toolset, allowing users to:

- describe a hardware network made up of IMS T9000 and IMS C104 packet routing switch devices
- define values for the user-programmable attributes of the devices (such as the IMS T9000's programmable memory interface, and the IMS C104 packet routing switch's interval labelling registers)
- define how processes in the application should be mapped to processors in the network

The tools allow the full user-programmable functionality of the IMS T9000 and IMS C104 devices to be used with simple textual descriptions of attribute values, without the need to write any low-level configuration code.

The configuration tools can check the network description to ensure that the network is properly connected. Applications of arbitrary connectivity can be mapped onto the hardware network. Where the network includes IMS C104 packet routing switches, the headers required to connect the channels between processors are calculated automatically by the tools. If the network does not contain IMS C104 packet routing switches, the tools provide software through-routing so that channels may still be connected between non-neighboring processors.

EPROM tools are provided to support the creation of initialization ROMs for IMS T9000 transputers, and the creation of system ROMs containing application code.

The debugger supplied in INQUEST supports source-level and low-level debugging of multi-process and multi-processor programs. The debugger runs on the host computer from which the IMS T9000 network has been loaded. A graphical user interface, with multiple windows, simplifies use of this sophisticated tool. INQUEST also includes profiling tools for detecting program 'hot-spots' and for evaluating how effectively parallel programs use a network of processors.

### 2.1.3 Hosts

Programs developed using the Toolset are both source and binary compatible across all host development machines. The INMOS T9000 ANSI C Toolset is available for the following development platforms:

IMS D4394 T9000 ANSI C Toolset for Sun-4 under SunOS 4.1.1

IMS D7394 T9000 ANSI C Toolset for IBM PC under MS-DOS 5

#### 2.1.4 Comparison with IMS Dx314 Professional ANSI C Toolset

The INMOS T9000 ANSI C Toolset is fully compatible with the IMS Dx314 Professional ANSI C Toolset. In particular:

- they are source compatible,
- the same libraries have been implemented to support parallelism,
- the parallel library interface is the same,
- the linker and librarian are functionally the same,
- the configuration language is upwards compatible.

In addition some new features have been introduced, including:

- a single compiler including all the optimizing features of the IMS Dx314 optimizing compiler
- a number of IMS T9000-specific optimizations,
- modified library code to better exploit the IMS T9000,
- new configuration tools for software and hardware configuration which support the new IMS T9000 communications architecture.

## 2.2 IMS T9000 networks

The new generation transputer, the IMS T9000, provides major enhancements to the communications capabilities of transputer networks. The INMOS T9000 ANSI C Toolset provides support for these features in the form of new configuration and initialization tools.

Using the IMS T9000 transputer and the INMOS T9000 ANSI C Toolset, a large network of processes and channels may be programmed quite independently of the transputer network on which it will be implemented. The IMS T9000 can multiplex a large number of channels, called virtual channels, onto a single physical link. This allows a large network of channels between processes to be efficiently implemented on small IMS T9000 networks without routing or multiplexing software.

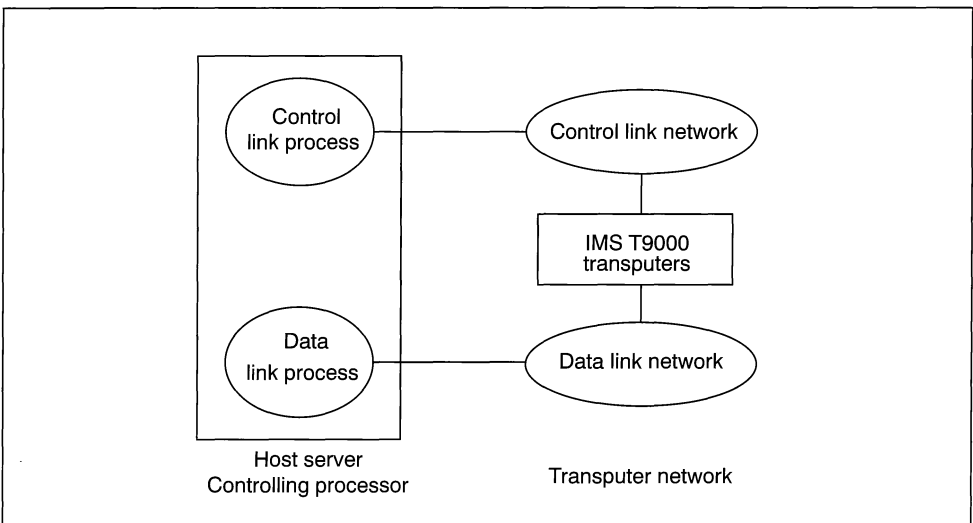


Figure 2.1 IMS T9000 network architecture

This hardware multiplexing support is complemented by a software through-routing mechanism provided by the INMOS T9000 ANSI C Toolset. The combination of hardware multiplexing and software through-routing allow any user-defined channel connections to be specified and implemented with optimal flexibility.

The IMS C104 packet routing switch gives higher communications performance in larger systems. It is a fast packet router which can route each incoming packet out along any one of its links, depending on the packet header added to the data by the sending transputer. It enables direct connections between the transputers in a large network and can remove the need for software through-routing. Large networks may also be implemented without packet routing switches, in which case each IMS T9000 may have a direct connection to up to four other devices.

An IMS T9000 transputer system has a network of links for data communications plus a separate network of control links for system initialization and debugging control. An IMS T9000 network can be viewed as a number of processors each of which is connected to both the data network and the control network. During system development, both networks should also be connected to the host or other controlling processor, as shown in Figure 2.1.

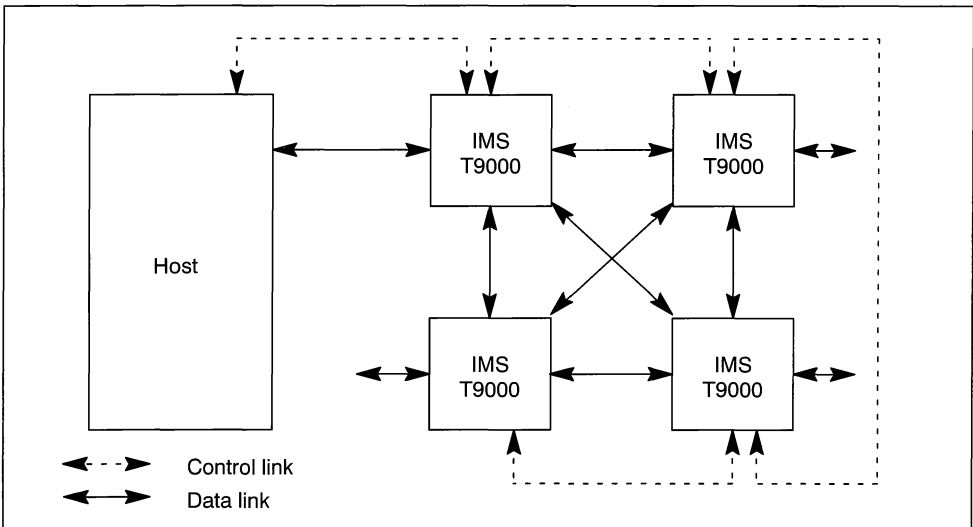


Figure 2.2 IMS T9000 network example

**2.2.1 The IMS T9000 family of devices**

The INMOS T9000 ANSI C Toolset supports the full family of the IMS T9000 transputer and associated communications and interface chips – the IMS C104 packet routing switch and the IMS C100 system protocol converter. The IMS C104 packet routing switch is a 32-way packet router. The IMS C100 system protocol converter allows a T9000 data link and control link to be connected to T2/T4/T8 links and system services. It provides a gateway to connect a T2/T4/T8 network to a IMS T9000 network.

**2.3 Parallel programming**

The INMOS T9000 ANSI C Toolset supports parallelism on individual transputers and parallelism across networks of IMS T9000 transputers.

The transputer programming model consists of parallel processes communicating through channels. Channels connect pairs of processes and allow data to be passed between them. Each process can itself be built from a number of parallel sub-processes, so that an entire software system can be described as

a process hierarchy. Processes may be created at high and low priority levels. Interrupt routines are typically implemented as high priority processes. This model is consistent with many modern software design methodologies.

Figure 2.4 shows a collection of four processes communicating through channels. The `mux` process communicates with a host computer and hands out work to be done to one of three `worker` processes. Results from the workers are then returned to the host by the `mux` process. The following example shows how this collection of processes can be described in C to run on a single IMS T9000 transputer. Section 2.6.4 shows how the parallel processes in the program can be mapped onto a network of processors using the configuration language.

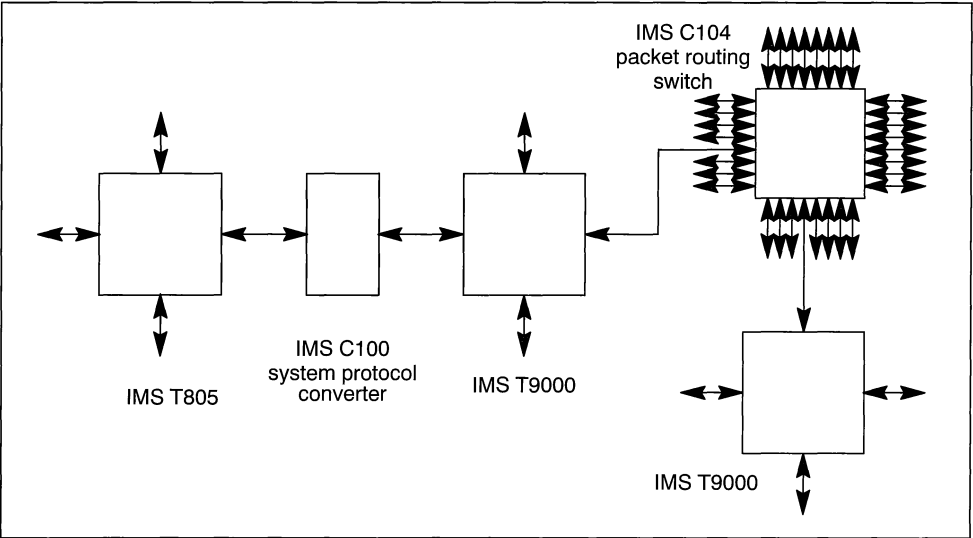


Figure 2.3 IMS T9000 family of devices

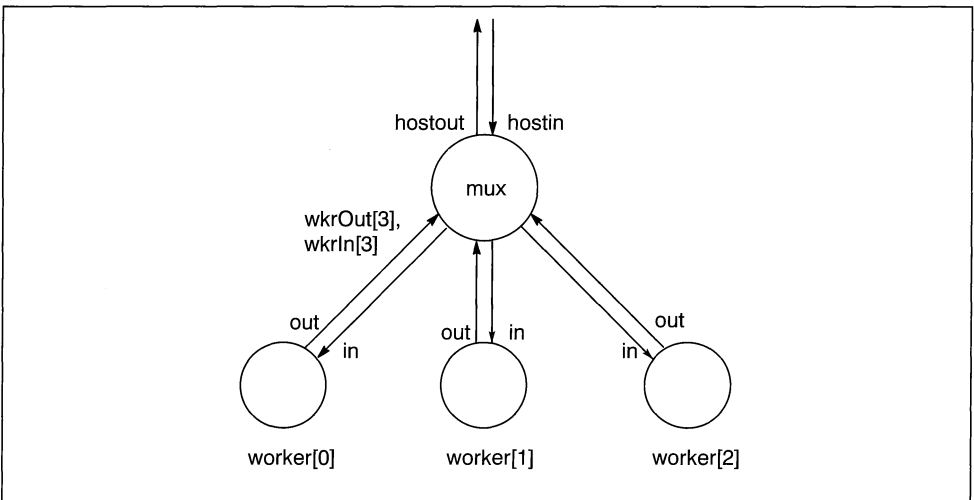


Figure 2.4 Software network



```

/* Example program */

#include <process.h>
#include <misc.h>
#include <stdlib.h>

/* Define prototypes for process functions */

void fmx (Process*, Channel*, Channel*, Channel*[], Channel*[], int);
/* process, hostin, hostout, in, out, no_workers */

void fwkr (Process*, Channel*, Channel*, int);
/* process, in, out, worker_id */

/* Main program */

int main (int argc, char *argv[], char *envp[],
          Channel *in[], int inlen, Channel *out[], int outlen)
{
    int i;

    /* Declare processes and channels */
    Channel *hostin, *hostout, *wkrIn[3], *wkrOut[3];
    Process *mux, *worker[3];

    /* Allocate channels */
    hostin = in[1];
    hostout = out[1];
    for (i = 0; i < 3; ++i) {
        wkrIn[i] = ChanAlloc ();
        wkrOut[i] = ChanAlloc ();
    }

    /* Allocate processes */
    mux = ProcAlloc (fmx, 0, 5, hostin, hostout, wkrIn, wkrOut, 3);
    for (i = 0; i < 3; ++i) {
        worker[i] = ProcAlloc (fwkr, 0, 3, wkrIn[i], wkrOut[i], i);
    }
    /* Start the processes running in parallel */
    ProcPar (mux, worker[0], worker[1], worker[2], NULL);

    exit (0);
}

```

Figure 2.5 Parallel processes

By using library calls, parallel processes may be created dynamically. Processes may be created individually in which case the function call will return immediately with the called process executing concurrently with the calling process, or created as a group, in which case the function will return when all the processes within the group have completed. Processes communicate by message passing over channels.

The example in Figure 2.5 illustrates how to program the collection of parallel processes shown diagrammatically in Figure 2.4. The functions `fmx` and `fwkr` contain the executable code of the processes `mx` and `worker` respectively. These processes communicate using channels.

The ANSI C compiler in the Toolset can be used to compile the program shown in Figure 2.5. The compiled code can then be linked, configured and collected to produce a code file to run on a single transputer.

Alternatively it may be preferred to distribute the processes over a network of more than one processor, in which case the code for the `mux` and `worker` processes would be compiled and linked separately. The linker produces processes in the form of fully linked units. A network of fully linked processes can be distributed over a network of processors using the configuration tools. The code shown in Figure 2.5 would be replaced by a configuration description, as described in Section 2.6.4.

Each process has its own stack space (typically allocated from the heap of the main process) but the processes in a single linked unit share static space, heap space and code (including libraries). Processes created in this way can communicate by message-passing over channels or by shared data (optionally protected by semaphores or channels).

Functions are provided to read a message from one of a list of channels (implementing the OCCam **ALT** construct), to time-out on channel input and to access the high and low resolution timers built into the IMS T9000 transputer. The transputer's hardware scheduler provides extremely efficient scheduling of these processes and efficiently implements many features which would normally require a real-time executive.

The ANSI C compiler operates from a host command line interface. The pre-processor is integrated into the compiler for fast execution. The compile time diagnostics provided by the compiler are excellent. These include type checking in expressions and type checking of function arguments.

The tools integrate into the host operating system build utilities, allowing, for example, the use of standard editor, make and source code and configuration control utilities.

## 2.4 Compiler and compilation tools

### 2.4.1 ANSI C conformance

The INMOS T9000 ANSI C Toolset supports the full standard language as defined in X3.159-1989. The key extensions and functions which are in the ANSI standard but not in the original definition of C by Kernighan and Ritchie are:

- Prototypes to define function parameters.
- Better definition of the pre-processor.
- Longer identifiers. The standard specifies at least 31 characters are significant in identifiers and six characters for external names. The INMOS ANSI C Toolset implementation allows arbitrarily long identifiers, the first 256 characters of which are significant.
- Trigraphs introduced for use by some character sets which do not have some of the normal C characters.
- Data items which can be declared as **const** or **volatile**.
- Support for special character sets (including Asian ones).
- Enumeration types.
- Implementation limits which are accessible from header files.
- Structures which can be assigned, passed to functions and returned from them.

The compiler passes all the tests in the validation suite from Plum Hall. The compiler will be validated by the British Standards Institution (BSI). The validation is recognized across Europe by the French (AFNOR) and Italian (IMQ) Standards Institutes. The standards authority in Japan (JMI) has also agreed to harmonize their C validation with Europe. The USA National Institute of Standards and Technology (NIST) is expected to recognize the validation in due course.

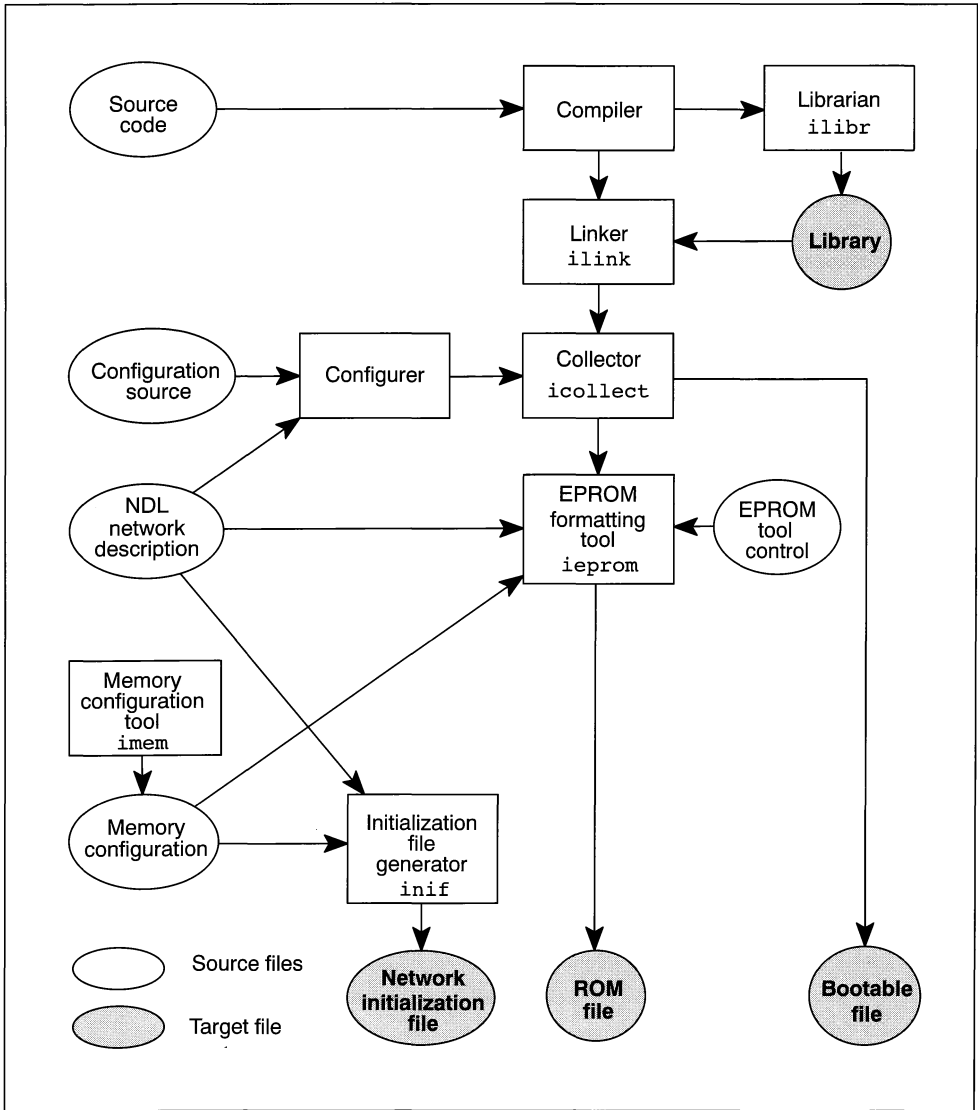


Figure 2.6 Program build model for the INMOS T9000 ANSI C Toolset

### 2.4.2 Optimization

The compiler implements a wide range of local code optimization techniques.

**Constant folding.** The compiler evaluates all integer and real constant expressions at compile time.

**Advanced workspace allocation.** Frequently used variables are placed at small offsets in workspace, thus reducing the size of the instructions needed to access them and ensuring that the IMS T9000's workspace cache is employed for frequently used variables. This therefore increases the speed of execution.

**Dead-code elimination.** Code that cannot be reached during the execution of the program is removed.

**Peep-hole optimization.** Code sequences are selected that are the fastest for the operation. For example, single precision floating variables are moved using the integer move operations.

**Instruction scheduling.** Where possible the compiler exploits the internal concurrency of the transputer.

**Constant caching.** Some constants have their load time reduced by placing them in a constant table.

**Unnecessary jumps** are eliminated.

**Switch statements.** The compiler can generate a number of different code sequences tailored to cover the dense ranges within the total range.

**Special idioms** that are better on transputers are chosen for some code sequences.

### Globally optimized code generation

When global optimizations is selected, the compiler extends the types of optimizations it performs to global techniques. These have typically given a 15%-25% improvement in speed over the local optimizations, as measured by a suite of internal benchmarks.

**Common sub-expression elimination** removes the evaluation of an expression where it is known that the value has already been computed; the value is stored in a temporary local workspace. This improves the speed of a program and reduces code size.

**Loop invariant code motion** can move the position where an expression is evaluated from within a loop to outside it. If the expression contains no variables that are changed during the execution of a loop, then the expression can be evaluated just once before the loop is entered. By keeping the result in a temporary location, the speed of execution of the whole loop is increased.

**Tail-call optimization** reduces the number of calls and returns executed by a program. If the last operation of a function is to invoke another function and immediately return the value therefrom, then the compiler attempts to re-use the same workspace area by just jumping to (rather than calling) the lower level function. The called function then returns directly to where the upper level function was called from. In the case where the call is a recursive call to the same function, then the workspace is exactly the right size, and a saving is also made because the stack adjustment instructions are no longer needed either. This optimization saves time and total stack space.

**Workspace allocation by coloring** reduces the amount of workspace required by using a word for two variables when it can be determined that they are not both required at the same time. In addition the variables that are most frequently used are placed at lower offsets in workspace. This reduces the number of prefixes needed to access the variables and so reduces the total code size.

The optimizing compiler also implements a pragma, `IMS_nosideeffects`, whereby the user can indicate that a function does not have any side-effects on external or static variables. The optimizer can then use this information to make assumptions about what state can be changed by a function invocation and hence about the validity of any previously computed expressions.

To make the access to some of the transputer's instructions even more effective, a number of special library functions have been defined which the optimizing compiler can render as in-line code. This removes the overhead of a library call, but it also gives the optimizer more information on what the program is doing.

Normally, when the optimizer sees a function containing some assembler code, it must make very conservative assumptions about the effect the code has on its surroundings, e.g. on static variables and parameters. By using the functions defined to access the instructions, the optimizer knows exactly what the effects will be and can make the minimally correct assumptions for the side-effects of the code.

The transputer instructions that can be accessed in this way include block moves, channel input and output, bit manipulation, CRC computations and some scheduling operations.

### 2.4.3 Use of IMS T9000 features

Programs are compiled to run as L-processes (with a local trap handler) on the IMS T9000. A trap handler is set up to catch and handle program runtime errors. Signal handlers can be set up to handle certain run-time errors.

The code generated by the compiler, and in the supporting libraries, makes full use of the new features introduced to the transputer instruction set in the IMS T9000. In particular:

- The part-word support instructions are used to improve the handling of 8-bit and 16-bit integers.
- The new floating point instructions, *fpsqrt* and *fprem* are used.
- *volatile* variables are implemented using the device access instructions.
- The semaphore support in the runtime library is implemented using the new semaphore instructions.
- Care has been taken in the code generation and optimization to ensure that effective use is made of the IMS T9000's workspace cache and CPU pipeline.

Access to the full IMS T9000 instruction set is available through assembly inserts.

### 2.4.4 TCOFF

The binary code produced by INMOS T9000 ANSI C Toolset tools is in Transputer Common Object File Format (TCOFF). This allows integration with other TCOFF-based utilities, including the INMOS T9000 OCCam 2 Toolset and the INQUEST development environment.

### 2.4.5 Separate compilation

Collections of subprograms can be compiled separately using the INMOS ANSI C compiler and optionally combined into a library. The linker is used to combine separately compiled functions and procedures as a linked unit. A single copy of a linked unit cannot be distributed over more than one processor. The linker supports selective loading of library units.

### 2.4.6 Mixed language programming

The INMOS T9000 ANSI C and OCCam 2 compilers are fully compatible and allow simple mixing of languages. OCCam and C processes may be freely mixed when configuring a program for a single transputer or a network of transputers. Such processes will run in parallel and communicate using channels.

The INMOS T9000 ANSI C Toolset allows OCCam 2 procedures and single-valued OCCam 2 functions to be called from C just like other C functions. Pragmas are provided to tell the C compiler not to generate the hidden static link parameter (which is required by C but not by OCCam) and to change the external names of procedures and function, since OCCam names may not be legal C function names.

Similarly, the INMOS T9000 OCCam 2 Toolset supports calling C functions directly from OCCam. C functions which require access to static variables are supported.

### 2.4.7 Assembler code

The INMOS T9000 ANSI C Toolset provides a very powerful assembler insert facility. Assembler code can be written at any point in the source code to achieve direct access to transputer instructions for reasons of speed or code size. Full access is available to the IMS T9000 transputer instruction set and C program variables.

The assembler insert facility supports:

- Access to the full instruction set of the IMS T9000 transputer
- Symbolic access to automatic and static variables
- Pseudo-operations to load multi-word values into registers
- Loading results of expressions to registers using the pseudo-operations.
- Labels and jumps
- Directives for instruction sizing, stack access, return address access etc.
- The INMOS ANSI C pre-processor may be used to provide macro assembly programming facilities.

If there is no other way to obtain the code required, for example when writing customized bootstrap mechanisms, then the final stage of the compiler may be invoked as an assembler to assemble user-written assembler code.

## 2.5 Libraries

The full set of ANSI C libraries is provided.

The standard mathematics library operates in double precision. Versions of the mathematical functions are provided that operate on float arguments and return float values. These libraries provide improved performance for applications where performance requirements override accuracy requirements.

The standard C mathematical functions provided use the same code as the OCCAM libraries. This ensures identical results and accuracy for all compilers supported by INMOS. A reduced C library is supplied to minimize code size for embedded systems applications and for processes which do not need to access host operating system facilities.

Extra libraries are provided to support parallel programming and to provide access to the special features of the transputer. Functions are provided to create parallel processes dynamically. Processes may be created individually in which case the function call will return immediately with the called process executing concurrently with the calling process, or created as a group, in which case the function will return when all the processes within the group have completed.

Functions are provided to support communications between processes by message passing over channels. Any form of data may be sent or received on a single channel. Input from a list of channels is also supported. Library functions provide access to the high and low resolution timers built into the transputer, allowing any process to read the current timer value or delay for a specified time. Channel inputs may also be timed out. Support is also provided for semaphores.

Support is given for loading code dynamically. Miscellaneous library functions are also provided, including functions to identify the type of host, the host link and the boot link.

## 2.6 Configuration, initialization and loading

Configuration is the process of defining how an application program is to be run on the available hardware. Given a description of the hardware network, the software network of an application and the mapping between them, the INMOS T9000 ANSI C Toolset produces a *bootable file* which can be sent to the network for execution. The hardware network description is also used to prepare a *network initialization file* which is used to initialize the network and bootstrap the processors into a state where they are ready to receive the bootable file.

A configuration description file is used as input to a tool known as the *configurer*. The description is written in a C-like language which is upwards compatible with the configuration language in the Dx314 Profes-

sional ANSI C Toolset. The configuration description describes the application as a network of processes and channels. It indicates which linked files should be used as the code for each process, and it also indicates on which processor each process is to be run.

The hardware description is separate from the rest of the configuration description. It is written in the INMOS Network Description Language (NDL). The configuration description refers to the NDL description of the hardware by means of a `#network` directive. The hardware description describes the processors and routing devices in the network and their connections.

The main features of the configuration tools are as follows:

- To initialize, boot and load an IMS T9000 network, it is not necessary to write any low-level code; instead the attributes of devices are specified by simple textual statements in the network description, and the low level code is generated automatically by the tools.
- The network description for a particular network can be written once, when the hardware is designed, and does not normally need to be changed between different applications.
- The configuration tools will check the correctness of the network description; this includes checking the labelling of the routing chips for deadlock freedom.
- Channels specified by the user in the configuration description are automatically mapped by the tools. In a system with routing devices, the configuration tools calculate the routing headers required for the user's channels. If there are no routing devices in the system, the configurer adds any software through-routing mechanisms required. Thus the software network is not constrained by the topology of the hardware network.
- For a particular application, the tools can produce either a bootable file which can be used to load the network from a host, or a boot-from-ROM file which can be used to program a ROM on one of the processors in the network.
- Initialization of individual IMS T9000 devices in the network (for example, setting up the external memory interface) can be done over the control link, or from an initialization ROM local to each IMS T9000. Support for generation of local initialization ROMs is included.

### 2.6.1 Network description

The Network Description Language (NDL) is used to describe the available hardware – the types of processors, their attributes and how they are connected. Processor attributes include, for example, a description of its memory map and its link speeds. The NDL also describes any packet routing switch devices in the network and their attributes. Attributes of a routing device include the *labelling* of the routing device, which indicates how packets from processors should be routed through it.

The network description will not normally change unless the hardware is changed. This NDL description of the system is used by the tools for a variety of purposes, from initializing the hardware to mapping application code onto processors. These tools can either read and check the NDL source directly or read a binary version produced by the NDL compiler, `ind1`.

Figure 2.7 shows the NDL network description for the example network shown in Figure 2.10.

### 2.6.2 Memory configuration

The IMS T9000 programmable memory interface supports a wide variety of memory configurations. It can be set up to provide the appropriate signals and timing needed by the memory being used. The parameters to initialize the memory interface may be sent to the transputer using the control link or may be included in the bootstrap code in ROM.

The memory configuration tool, `imem`, is used to assess and define memory interface parameters for each IMS T9000. It can be run interactively or in batch mode. The output may be incorporated into the network

initialization file or into a ROM. The memory configuration description will not normally change unless the hardware is changed. imem can also be used to generate memory interface timing documentation.

```
#INCLUDE "stdndl.inc"

— Name of file with memory timing information, etc.
VAL memconfig IS "T9000.mem" :

VAL n          IS 4 :           — Number of transputers
VAL SysMem     IS 4*K :         — Size of system RAM
VAL UsrBase    IS #80000000 :   — Start address of user RAM
VAL UsrMem     IS (2*M) - SysMem : — Size of user RAM
VAL SysBase    IS UsrBase+UsrMem : — Start of system RAM
                                       — (= top of user RAM)
VAL memoryflags IS [[SysBase, SysMem, RAM + SYSTEM],
                   [UsrBase, UsrMem, RAM + USER]] :

[n]NODE p :                       — n transputers
CONTROLPORT host :
ARC          hostlink :

NETWORK fourT9
DO
  — set link speeds
  SET DEFAULT (link.speed.multiply := 10)
  SET DEFAULT (link.speed.divide   := [1])
  SET DEFAULT (control.speed.divide := [8])

  — set processor types
  DO i=0 FOR n
    SET p[i] (type := "T9000")
  SET p[0] (root := TRUE)
  DO i=1 FOR n-1
    SET p[i] (root := FALSE)

  — set memory information
  DO i=0 FOR n
    SET p[i] (memconfig, memory := memconfig, memoryflags)

  — connect control network
  CONNECT host[control] TO p[0][control.up]
  DO i=0 FOR n-1
    CONNECT p[i][control.down] TO p[i+1][control.up]

  — connect data links
  CONNECT host[data] TO p[0][link][0] WITH hostlink
  DO i=0 FOR n-1
    CONNECT p[i][link][1] TO p[i+1][link][2]
  CONNECT p[n-1][link][1] TO p[0][link][2]
  DO i=0 FOR n-2
    CONNECT p[i][link][3] TO p[i+2][link][0]
:

```

Figure 2.7 NDL network description example



### 2.6.3 Initializing and loading

A hosted IMS T9000 network is initialized by first sending control commands to the transputers and routers connected to the control network and then loading code using the data network.

The correct sequence of control network commands is held in a network initialization file, which contains all the information needed to initialize a system through the control links prior to loading the application code. It can be automatically generated by the initialization file generator tool, `inif`, from the network description and memory configurations. To initialize the network, the network initialization file is used by the initialization software to generate the correct sequence of commands to send to the control link network.

The bootstrap code for each processor, loading code and user application code are all incorporated in the bootable file with the necessary routing information. The bootable file is generated automatically by the collector, `icollect`, from the configuration information and linked application code files. To load the code onto the network, the bootable file is sent down the data link to the data link network.

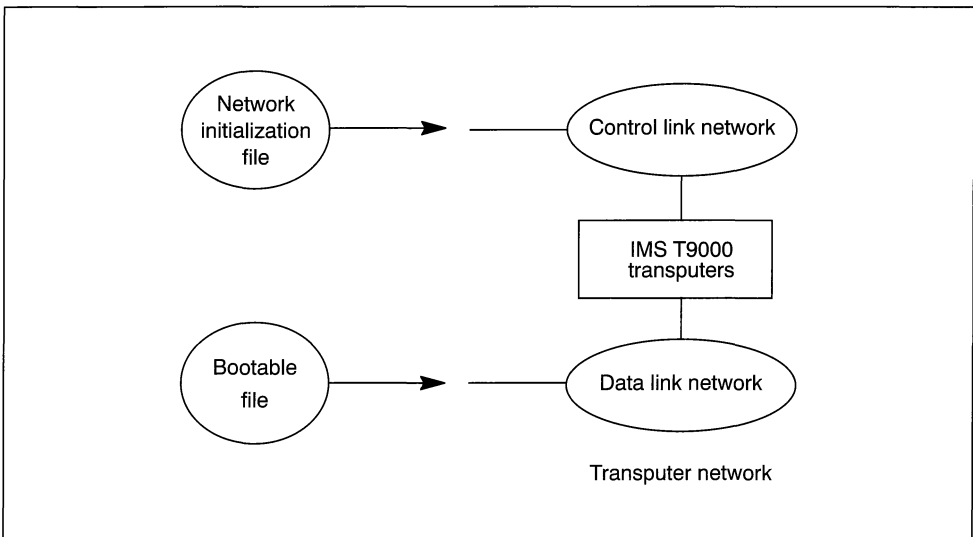


Figure 2.8 Initializing a hosted IMS T9000 network

### 2.6.4 Configuration language

A C-like configuration language is used to describe the network of processes and channels and the mapping of the processes onto the transputer network. The transputer network is described separately in the network description file. Multiple processes may be mapped onto the same transputer. The network description file is referred to by means of a `#network` configurer directive. This allows the user to map processes in the configuration description onto the processors named in the network description file. The routing of channels may be generated automatically or may be included in the configuration description.

The following example illustrates just how easy it is to configure a program for transputers. Instead of using the top level single transputer code shown in Figure 2.5, it may be preferred to distribute the program over a network of processors, as shown in this example.

```

/* Configuration example */

/* Hardware description */

#network "fourT9.ndl"

/* Software description */

/* Define process memory sizes and interfaces */
process (stacksize = 2K, heapsize = 16k); /* Define defaults */
rep i = 0 for 3
    process (interface (input in, output out, int id)) wkr[i];
process (interface (input hostin, output hostout,
                    input in[3], output out[3])) mux;

/* Define external channels, interconnections and parameters */
input hostinput; /* Host channel edges */
output hostoutput;
connect mux.hostin to hostinput; /* Host channel connections */
connect mux.hostout to hostoutput;
rep i = 0 for 3
{
    worker[i] (id = i); /* Set worker process id parameter*/
    connect mux.in[i] to wkr[i].out;
    connect mux.out[i] to wkr[i].in;
}

/* Mapping description */

/* Define linked file units for processes */
use "mux.lku" for mux;
rep i = 0 for 3
    use "wkr.lku" for worker[i];

/* Map processes to processors and external channels to edges */
rep i = 0 for 3
    place worker[i] on p[i+1];
place mux on p[0];
place hostinput on host;
place hostoutput on host;

```

Figure 2.9 Configuration

Figure 2.9 shows the configuration text for describing the software network of processes and channels and mapping the software onto the hardware shown in Figure 2.10 and described in NDL in Figure 2.7. The `mux` and `worker` processes in the software network have been compiled and linked into the files `mux.lku` and `wkr.lku` respectively. The NDL shown in Figure 2.7 is in the file `fourT9.ndl`. The software description in the configuration text replaces the top level of code shown in Figure 2.5. In this example the `mux` process runs on the root transputer, called `p[0]`, while the individual `worker` processes run one on each of the transputers labelled `p[1]` to `p[3]` in Figure 2.10.

### 2.6.5 ROM support

The software can usually be developed and tested in a hosted development system without ROM by using host files loaded via transputer links. As a final stage, the completed application software and initialization data can be loaded into EPROMs using the EPROM program formatting tool, `ieprom`.

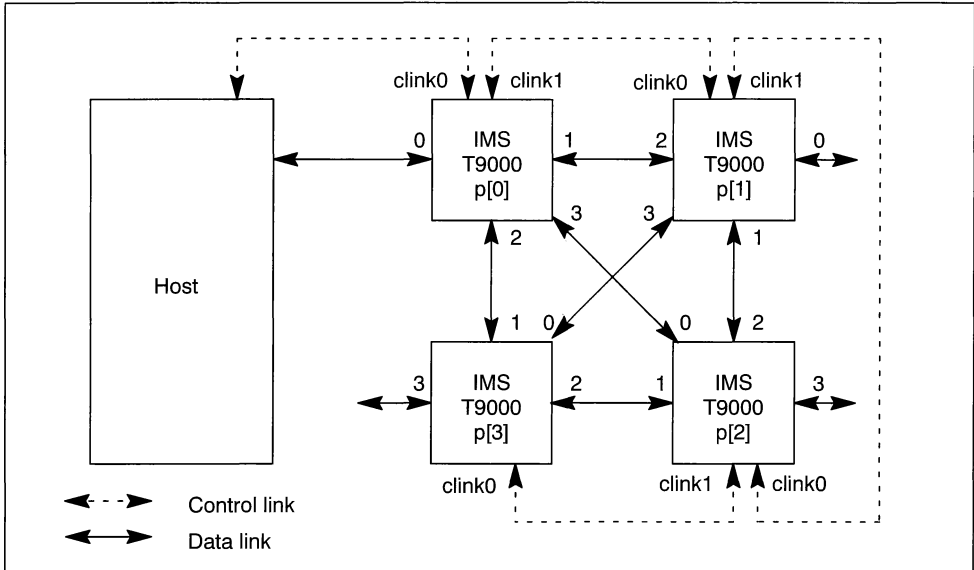


Figure 2.10 IMS T9000 network example showing link numbers

The EPROM formatting tool can produce a file suitable for programming a ROM; this may be either a system ROM or a local ROM. A system ROM holds network initialization data and application software for initializing and loading onto a network of transputers. Stand-alone systems will need to boot from a system ROM. A local ROM holds local initialization data for only one transputer in a network. One or more transputers in a network may be initialized from local ROMs, whether the application is loaded from a host file or from a system ROM.

The EPROM formatting tool may read the network description, memory configurations and the collected application software. From this data, the EPROM formatting tool can build a ROM file incorporating the network initialization data and application software. A control file allows the user to control how the data is arranged in the ROM. The ROM file may be produced in ASCII hexadecimal, Intel hexadecimal, extended Intel hexadecimal, Motorola S-record or binary format.

## 2.7 Connecting IMS T9000 networks to T2/T4/T8 networks

IMS T9000 transputers have improved physical links and data protocols from the T2/T4/T8 family of devices. In addition the method of initializing and controlling networks of devices is enhanced; IMS T9000 networks have a network of control links which is used to control (reset, bootstrap, monitor and analyze) devices in the network.

The IMS C100 is a protocol converter chip for connecting IMS T9000 and T2/T4/T8 networks together, and for converting the data communication and control interfaces. For detailed information on the IMS C100, please consult the appropriate data sheet.

It is envisaged that IMS T9000 users will adopt one of the following options, which are all supported by the software:

- 1 Use only T9000 family devices. Existing users of T2/T4/T8 networks can recompile existing T2/T4/T8 code onto a new IMS T9000 network.
- 2 Connect a T2/T4/T8 network to an T9000 family network, using an IMS C100 system protocol converter as a gateway between the networks, as shown in Figure 2.11. The T9000 ANSI C

Toolset should be used for the IMS T9000 network and the Dx314 Professional ANSI C Toolset should be used for the T2/T4/T8 network. The two networks can be loaded separately or either can be loaded by the other.

- Build an IMS T9000 network with T2/T4/T8 products as peripherals, using IMS C100 system protocol converters (in mode 2) as interfaces. The code for each T2/T4/T8 transputer should be compiled and linked using the Dx314 Professional ANSI C Toolset. The T2/T4/T8 transputers can be loaded by the IMS T9000 network.

When networks of different types are connected together (i.e. in options 2 and 3 above), the user can arrange for the networks to be loaded separately (from a host file or from ROM) or one network can load the other. The INMOS T9000 ANSI C Toolset includes support for initializing and loading one type of network from the other type, as follows:

- The IMS C100 system protocol converter can be used (in mode 1) to allow a T2/T4/T8 processor to control an IMS T9000 network (reset the network, initialize it and load it with an application). Software is provided in the INMOS T9000 ANSI C Toolset for inclusion in users' applications, to drive the IMS C100 from a T2/T4/T8 processor to perform these control functions.
- The IMS C100 system protocol converter can be used (in mode 2) to allow an IMS T9000 processor to control a T2/T4/T8 network (reset it and load it with an application). Software is provided in the INMOS T9000 ANSI C Toolset for inclusion in users' applications, to drive the IMS C100 from an IMS T9000 to perform these control functions.

The IMS C100 has two other modes (modes 0 and 3) which are not used by this toolset.

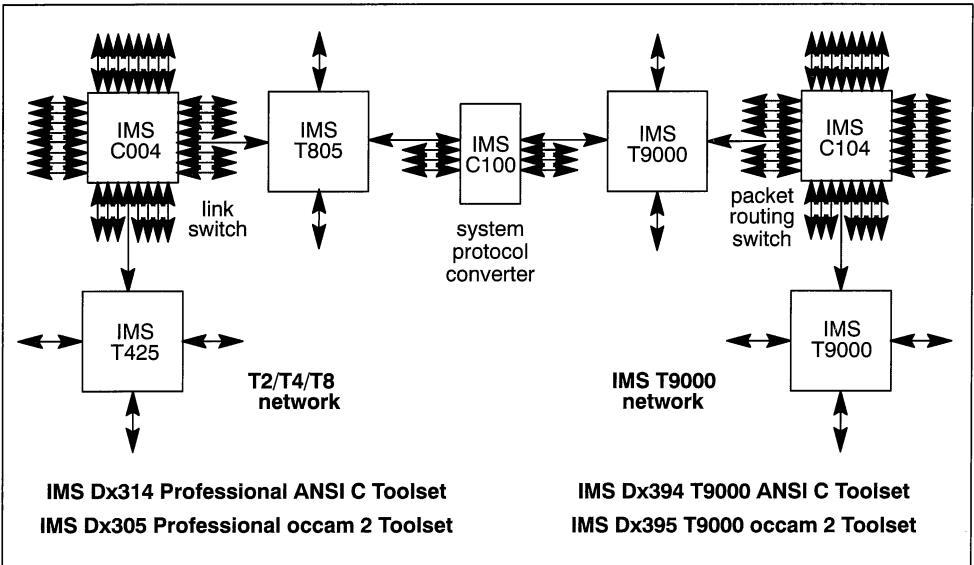


Figure 2.11 A mixed network example

## 2.8 Product components

### 2.8.1 Tools

ANSI C compiler, linker and librarian – `icc`, `ilink`, `ilibr`

Makefile generator, binary lister program and memory map lister – `imakef`, `ilist`, `imap`

Configuration tools – `indl`, `imem`, `inif`, `iconf`, `icollect`  
EPROM programming tool – `ieprom`

### 2.8.2 Libraries

Full ANSI library plus parallel support – `libc.lib`  
Reduced library for embedded systems – `libcred.lib`  
Mixed language support library – `centry.lib`  
Configuration support libraries

### 2.8.3 Sources

Programming examples  
Makefile generator source files  
Configuration support library source files  
Network description examples  
C start-up source files  
Network control software source files

### 2.8.4 Documentation

Toolset User Guide  
Tools Reference Manual  
Language and Libraries Reference Manual  
System Configuration Guide  
Delivery Manual  
ANSI C Toolset Handbook

## 2.9 Product variants

### 2.9.1 Sun-4 product

#### Product

- IMS D4394 T9000 ANSI C Toolset

#### Operating requirements

For Sun-4 hosted cross-development the following will be required:

- A Sun-4 workstation or server
- SunOS 4.1.1 or later
- 10 Mbytes of free disk space.

For loading target systems, a suitable transputer network interface will be required.

#### Distribution media

Sun-4 software is distributed on DC600A data cartridges 60 Mbytes, QIC-24, tar format.

## Licensing

The IMS D4394 T9000 ANSI C Toolset is a four-user product. For each product purchased, up to four users are able to use the Toolset concurrently at any one customer site. The tools can be run on any Sun-4 machine that is part of a network connected to a single machine where the licence manager is installed. Further information about the licence manager is included in the product Delivery Manual. Multiple copies can be purchased for larger project teams using volume discount curves.

No licence fee is charged for including INMOS libraries in customer products when linked with customer applications using the INMOS linker, `ilink`. Example programs and other sources provided may be included in software products, but INMOS Limited retain original copyright. Full licensing details are available from SGS-THOMSON Sales Offices, Regional Technology Centers and authorized distributors.

### 2.9.2 IBM product

#### Product

- IMS D7394 T9000 ANSI C Toolset

#### Operating requirements

For PC hosted cross-development one of the following will be required:

- IBM PC with a 386 or 486 processor and a minimum of 4 Mbytes memory

In each case the following will be required:

- DOS 5.0 or later
- 9 Mbytes of free disk space

For loading target systems, a suitable transputer network interface will be required.

#### Distribution media

Software is distributed on two media systems, both of which are supplied in the product:

- 1.2 Mbytes (96TPI) 5.25 inch IBM format floppy disks
- 1.44 Mbytes 3.5 inch IBM format floppy disks.

## Licensing

The IMS D7394 T9000 ANSI C Toolset is a single-user product. Multiple copies can be purchased for larger project teams using volume discount curves.

No licence fee is charged for including INMOS libraries in customer products when linked with customer applications using the INMOS linker, `ilink`. Example programs and other sources provided may be included in software products, but INMOS Limited retain original copyright. Full licensing details are available from SGS-THOMSON Sales Offices, Regional Technology Centers and authorized distributors.

## 2.10 Problem reporting and field support

A registration form is provided with each product. Return of the registration form will ensure you are eligible for future product updates. Software problem report forms are included with the software. INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centers and authorized distributors.

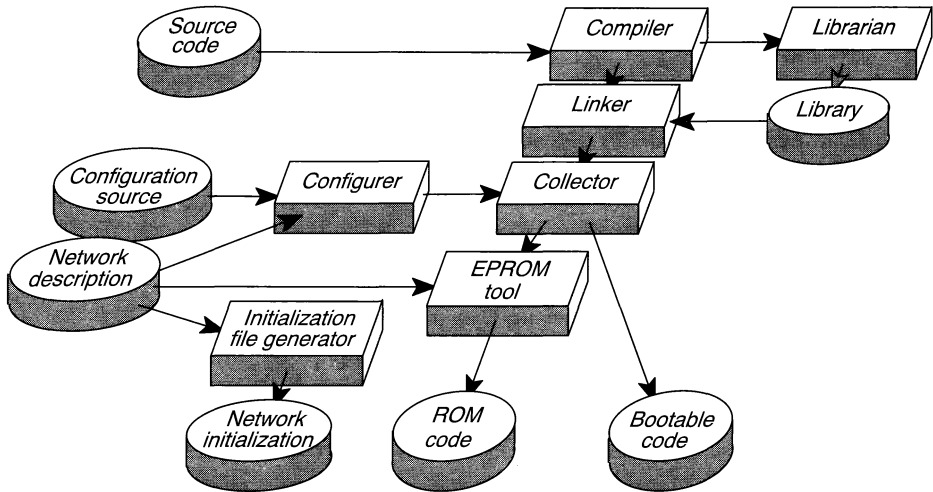


# IMS Dx395

## T9000 occam 2 Toolset

### Preliminary Information

The information in this datasheet is subject to change.



#### KEY FEATURES

- Complete OCCam 2 toolset for IMS T9000 transputer networks
- Source compatible with current INMOS development tools
- Code generation for IMS T9000 instruction set and pipelined CPU
- Configuration tools that exploit new communications architecture
- Software routing used for networks without routing chips
- Support for mixing ANSI C and OCCam 2
- Support for communication between T2/T4/T8 networks and IMS T9000 networks
- Support for assembler inserts
- Support for EPROM programming

#### DESCRIPTION

The INMOS T9000 OCCam 2 Toolset supports the construction of OCCam 2 programs which may be loaded onto IMS T9000 networks via a link, or put into a form suitable for booting from ROM. The Toolset additionally includes the hardware configuration tools required to initialize IMS T9000 networks, fully supporting the use of IMS C104 packet routing switches including initialization and labelling of networks. The toolset supports the use of the IMS C100 system protocol converter device which provides a means to connect IMS T2xx/T4xx/T8xx networks to IMS T9000 networks.

## 3.1 Introduction

This document contains advance information for the INMOS T9000 occam 2 Toolset. Within this document the current IMS T2xx, T4xx and T8xx transputers are called 'T2/T4/T8' transputers.

The INMOS T9000 OCCam 2 Toolset provides a complete OCCam 2 cross-development system for IMS T9000 transputer network targets. It can be used to build sequential or parallel programs for single IMS T9000 transputers or for multi-transputer IMS T9000 networks.

The Toolset supports the construction of OCCam 2 programs, which may be loaded onto IMS T9000 networks via a link or put into a form suitable for booting from ROM. The Toolset also includes the hardware configuration tools required to initialize IMS T9000 networks. It fully supports the use of IMS C104 packet routing switches including labelling and initializing. The Toolset also supports the use of the IMS C100 system protocol converter device which provides a means to connect T2/T4/T8 networks to IMS T9000 networks.

### 3.1.1 Key features

- OCCam 2 cross-compiler for IMS T9000 target networks
- Code generation for IMS T9000 instruction set and pipelined CPU
- Scientific libraries
- Support for assembler inserts
- Listings of where variables and functions reside in memory
- Separate compilation, using linker and librarian tools
- Tools for creating and loading multi-processor programs
- Exploitation of the new communications architecture by the configuration tools
- Software routing of channels for networks without routing chips
- Automatic makefile generator
- Mixed language programming support
- Tools to support preparation of programs for EPROM
- Consistent tools across PC and Sun-4 hosts

### 3.1.2 Overview

The IMS T9000 transputer range of devices will be supported by the following software products:

- IMS Dx394 T9000 ANSI C Toolset
- IMS Dx395 T9000 OCCam 2 Toolset
- IMS Dx390 T9000 INQUEST development environment, containing debugging and profiling tools

In addition there will be other software products supporting the loading, running and serving of programs over specific host / IMS T9000 interface boards. This software will be supplied with the hardware.

The INMOS T9000 OCCam 2 Toolset is source compatible with INMOS T2/T4/T8 Toolset products. This has been achieved by using similar compiler, libraries, linker, configurer and other tools. Code written for a single IMS T4xx or IMS T8xx can be ported directly to an IMS T9000 for performance enhancement.

The IMS T9000 offers two important enhancements; the new communications architecture and the new pipelined CPU. In order to take full advantage of these features, a new Toolset has been developed which



will function alongside the existing Dx395 T9000 OCCam 2 Toolset. It is envisaged that IMS T9000 users will adopt one of the following options, which are all supported by the software:

- Use only IMS T9000 family devices. Users of T2/T4/T8 networks can recompile existing T2/T4/T8 code for a new IMS T9000 network.
- Connect a T2/T4/T8 network to an IMS T9000 family network, using an IMS C100 system protocol converter as a gateway between the networks.
- Build an IMS T9000 network with T2/T4/T8 products as peripherals, using IMS C100 system protocol converters as interfaces.

The configuration tools are the main area in which the difference between the IMS T9000 and the T2/T4/T8 range is apparent to the programmer. *Configuration* is the process of mapping a multi-process network application to a transputer network. Tools are provided in the INMOS T9000 OCCam 2 Toolset, allowing users to:

- describe a hardware network made up of IMS T9000 and IMS C104 packet routing switch devices
- define values for the user-programmable attributes of the devices (such as the IMS T9000's programmable memory interface, and the IMS C104 packet routing switch's interval labelling registers)
- define how processes in the application should be mapped to processors in the network

The tools allow the full user-programmable functionality of the IMS T9000 and IMS C104 devices to be used with simple textual descriptions of attribute values, without the need to write any low-level configuration code.

The configuration tools can check the network description to ensure that the network is properly connected. Applications of arbitrary connectivity can be mapped onto the hardware network. Where the network includes IMS C104 packet routing switches, the headers required to connect the channels between processors are calculated automatically by the tools. If the network does not contain IMS C104 packet routing switches, the tools provide software through-routing so that channels may still be connected between non-neighboring processors.

EPROM tools are provided to support the creation of initialization ROMs for IMS T9000 transputers, and the creation of system ROMs containing application code.

The debugger supplied in INQUEST supports source-level and low-level debugging of multi-process and multi-processor programs. The debugger runs on the host computer from which the IMS T9000 network has been loaded. A graphical user interface, with multiple windows, simplifies use of this sophisticated tool. INQUEST also includes profiling tools for detecting program 'hot-spots' and for evaluating how effectively parallel programs use a network of processors.

### 3.1.3 Hosts

Programs developed using the Toolset are both source and binary compatible across all host development machines. The INMOS T9000 OCCam 2 Toolset is available for the following development platforms:

IMS D4395 T9000 OCCam 2 Toolset for Sun-4 under SunOS 4.1.1

IMS D7395 T9000 OCCam 2 Toolset for IBM PC under MS-DOS 5

### 3.1.4 Comparison with IMS Dx305 Professional OCCam 2 Toolset

The INMOS T9000 OCCam 2 Toolset is fully compatible with the IMS Dx305 Professional OCCam 2 Toolset. In particular:

- they are source compatible,
- the same libraries have been implemented,

- the library interface is the same,
- the linker and librarian are functionally the same,
- the configuration language is upwards compatible.

In addition some new features have been introduced, including:

- a number of IMS T9000-specific optimizations,
- modified library code to better exploit the IMS T9000,
- new configuration tools for software and hardware configuration which support the new IMS T9000 communications architecture.

### 3.2 IMS T9000 networks

The new generation transputer, the IMS T9000, provides major enhancements to the communications capabilities of transputer networks. The INMOS T9000 occam 2 Toolset provides support for these features in the form of new configuration and initialization tools.

Using the IMS T9000 transputer and the INMOS T9000 occam 2 Toolset, a large network of processes and channels may be programmed quite independently of the transputer network on which it will be implemented. The IMS T9000 can multiplex a large number of channels, called virtual channels, onto a single physical link. This allows a large network of channels between processes to be efficiently implemented on small IMS T9000 networks without routing or multiplexing software.

This hardware multiplexing support is complemented by a software through-routing mechanism provided by the INMOS T9000 occam 2 Toolset. The combination of hardware multiplexing and software through-routing allow any user-defined channel connections to be specified and implemented with optimal flexibility.

The IMS C104 packet routing switch gives higher communications performance in larger systems. It is a fast packet router which can route each incoming packet out along any one of its links, depending on the packet header added to the data by the sending transputer. It enables direct connections between the transputers in a large network and can remove the need for software through-routing. Large networks may also be implemented without packet routing switches, in which case each IMS T9000 may have a direct connection to up to four other devices.

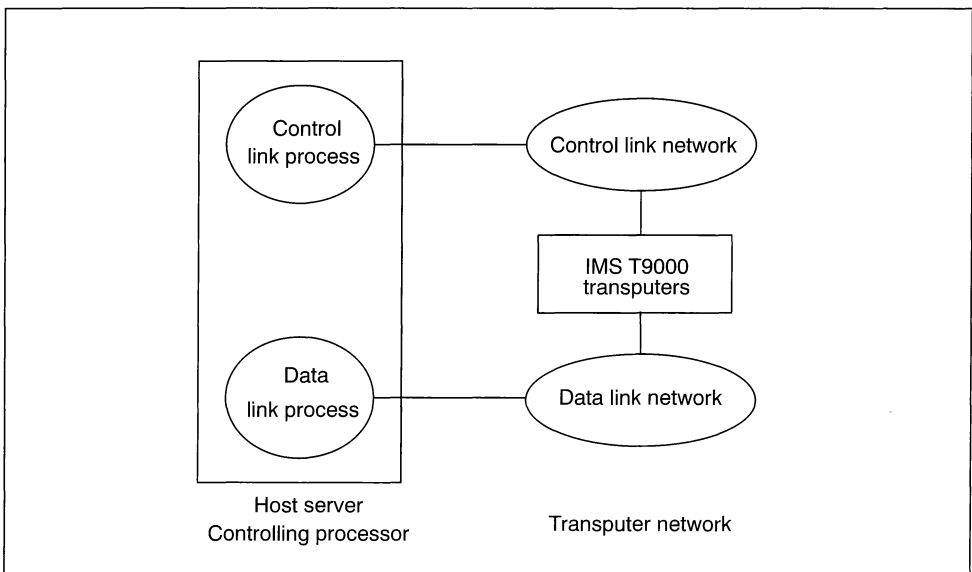


Figure 3.1 IMS T9000 network architecture

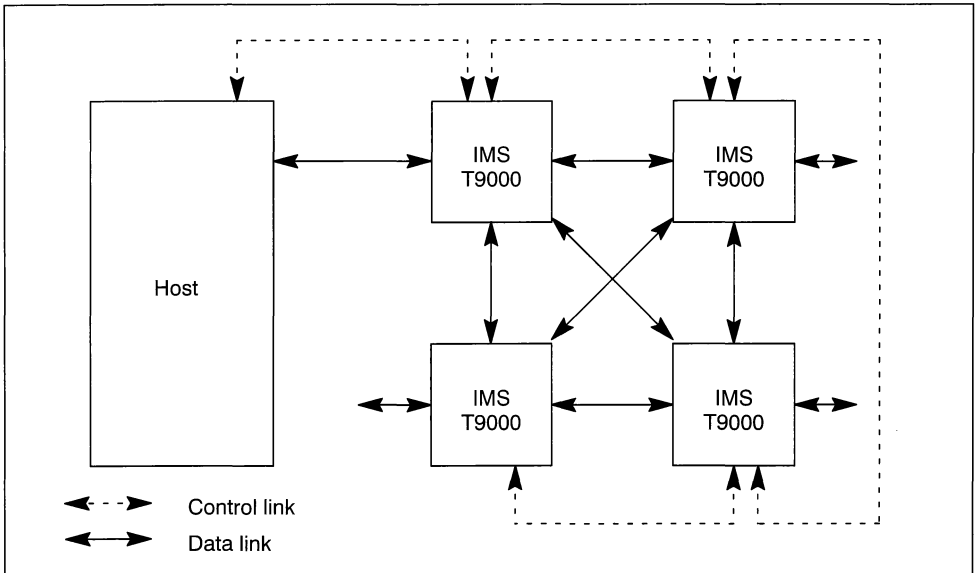


Figure 3.2 IMS T9000 network example

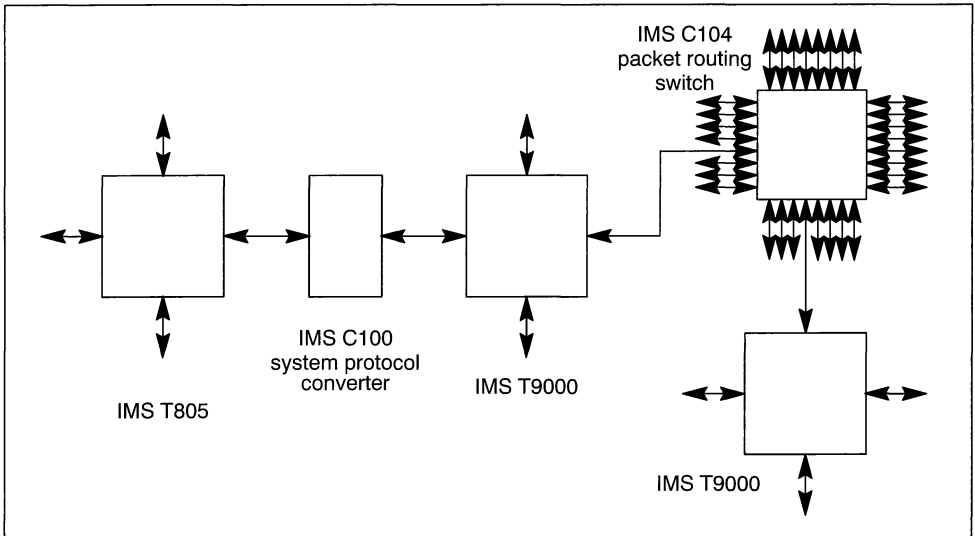


Figure 3.3 IMS T9000 family of devices

An IMS T9000 transputer system has a network of links for data communications plus a separate network of control links for system initialization and debugging control. An IMS T9000 network can be viewed as a number of processors each of which is connected to both the data network and the control network. During system development, both networks should also be connected to the host or other controlling processor, as shown in Figure 2.1.

### 3.2.1 The IMS T9000 family of devices

The INMOS T9000 occam 2 Toolset supports the full family of the IMS T9000 transputer and associated communications and interface chips – the IMS C104 packet routing switch and the IMS C100 system

protocol converter. The IMS C104 packet routing switch is a 32-way packet router. The IMS C100 system protocol converter allows a T9000 data link and control link to be connected to T2/T4/T8 links and system services. It provides a gateway to connect a T2/T4/T8 network to a IMS T9000 network.

### 3.3 Parallel programming

The INMOS T9000 *occam 2* Toolset supports parallelism on individual transputers and parallelism across networks of IMS T9000 transputers.

The transputer programming model consists of parallel processes communicating through channels. Channels connect pairs of processes and allow data to be passed between them. Each process can itself be built from a number of parallel sub-processes, so that an entire software system can be described as a process hierarchy. Processes may be created at high and low priority levels. Interrupt routines are typically implemented as high priority processes. This model is consistent with many modern software design methodologies.

*occam 2* is an efficient and effective language for parallel programming. It is a high level, structured language, providing simple constructs for creating parallel processes and for message-passing using channels. Each channel has a *PROTOCOL* which determines the types of messages that may flow along it. The *occam* compiler provides comprehensive compile and run-time checking which considerably reduces the number of coding and usage errors.

Figure 2.4 shows a collection of four processes communicating through channels. The *mux* process communicates with a host computer and hands out work to be done to one of three *worker* processes. Results from the workers are then returned to the host by the *mux* process. The following example shows how this collection of processes can be described in *occam* to run on a single IMS T9000 transputer. Section 2.6.4 shows how the parallel processes in the program can be mapped onto a network of processors using the configuration language.

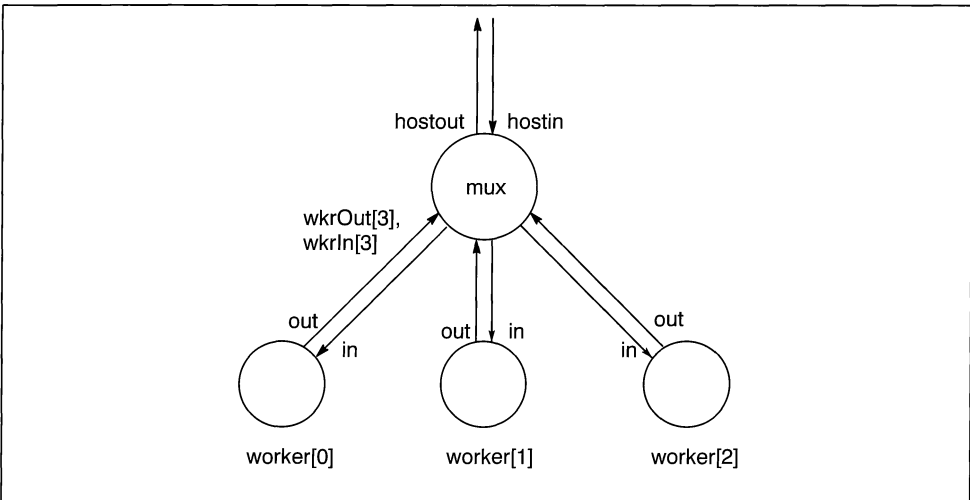


Figure 3.4 Software network

The example in Figure 2.5 illustrates how to program the collection of parallel processes shown diagrammatically in Figure 2.4. The files *mux.tco* and *worker.tco* contain the executable code of the processes *mux* and *worker* respectively. These processes communicate using channels.

The *occam 2* compiler in the Toolset can be used to compile the program shown in Figure 2.5. The compiled code can then be linked, configured and collected to produce a code file to run on a single transputer. Alternatively it may be preferred to distribute the processes over a network of more than one processor, in which case the code for the *mux* and *worker* processes would be compiled and linked separately. The linker produces processes in the form of fully linked units. A network of fully linked processes

can be distributed over a network of processors using the configuration tools. The code shown in Figure 2.5 would be replaced by a configuration description, as described in Section 2.6.4.

```
#INCLUDE "hostio.inc" — contains SP protocol definition
#USE "mux.tco"
#USE "worker.tco"

PROC example (CHAN OF SP hostinput, hostoutput, [ ]INT memory)
  [3]CHAN OF SP wkrIn, wkrOut:
  PAR
    mux(hostinput, hostoutput, wkrOut, wkrIn)
  PAR i = 0 FOR 3
    worker(wkrIn[i], wkrOut[i])
  :
```

Figure 3.5 Parallel processes

OCCAM constructs are provided to read a message from one of a list of channels, to time-out on channel input and to access the high and low resolution timers built into the IMS T9000 transputer. The transputer's hardware scheduler provides extremely efficient scheduling of these processes and efficiently implements many features which would normally require a real-time executive.

The tools integrate into the host operating system build utilities, allowing, for example, the use of standard editor, make and source code and configuration control utilities.

## 3.4 Compiler and compilation tools

### 3.4.1 Optimization

The compiler implements a wide range of code optimization techniques.

**Constant folding.** The compiler evaluates all constant expressions at compile time.

**Advanced workspace allocation.** Frequently used variables are placed at small offsets in workspace, thus reducing the size of the instructions needed to access them and ensuring that the IMS T9000's workspace cache is employed for frequently used variables. This therefore increases the speed of execution.

**Dead-code elimination.** Code that cannot be reached during the execution of the program is removed.

**Peep-hole optimization.** Code sequences are selected that are the fastest for the operation.

**Instruction scheduling.** Where possible the compiler exploits the internal concurrency of the transputer.

**Constant caching.** Some constants have their load time reduced by placing them in a constant table.

**Special idioms** that are better on transputers are chosen for some code sequences.

**In-line code** Procedures and functions can optionally be implemented as in-line code.

The compiler and linker provide features which allow programmers to make good use of the IMS T9000 transputer's internal memory when used as on-chip RAM.

### 3.4.2 Use of IMS T9000 features

Programs are compiled to run as L-processes (with a local trap handler) on the IMS T9000. A trap handler is set up to catch and handle program runtime errors. The behavior of the program after an error (HALT or STOP) is determined by the trap handler installed rather than at compilation time.

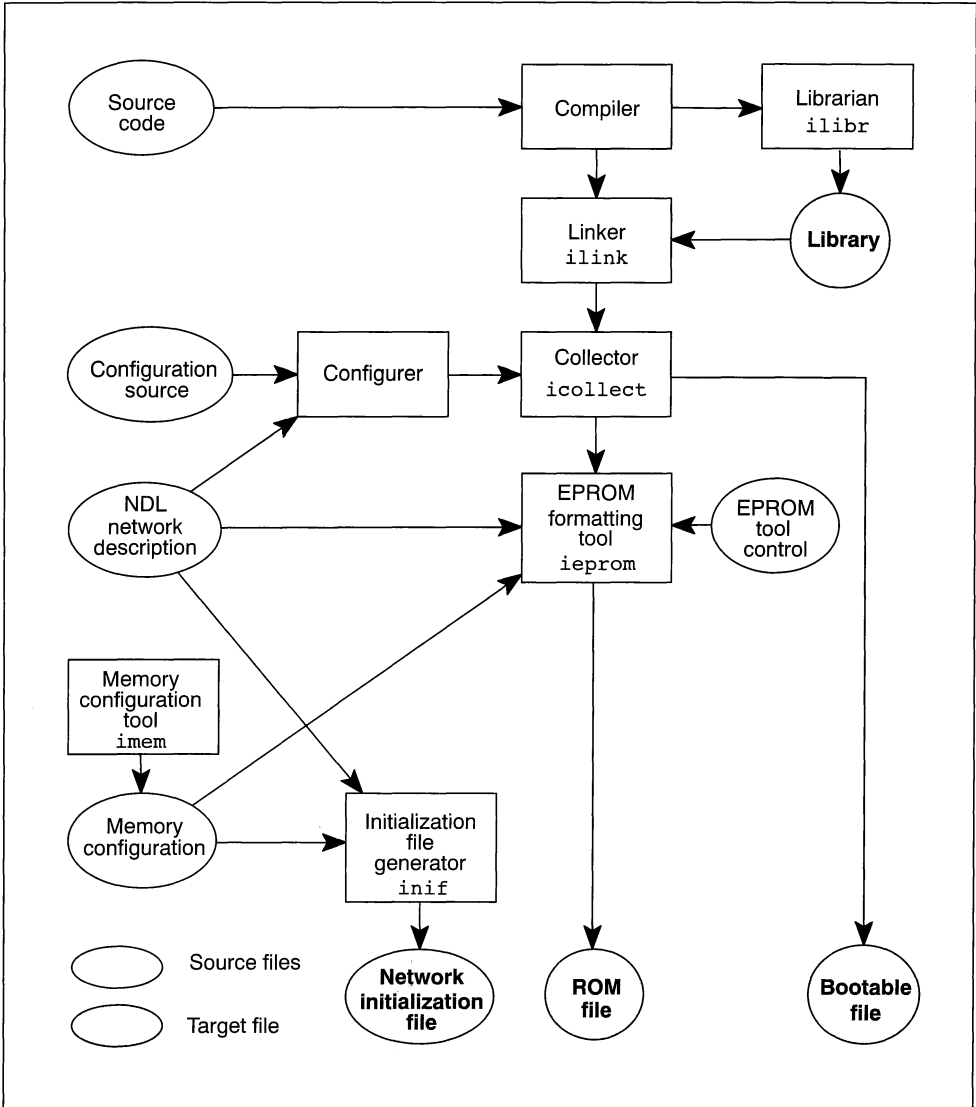


Figure 3.6 Program build model for the INMOS T9000 occam 2 Toolset

The code generated by the compiler, and in the supporting libraries, makes full use of the new features introduced to the transputer instruction set in the IMS T9000. In particular:

- The part-word support instructions are used to improve the handling of 8-bit and 16-bit integers.
- The new floating point instructions, *fpsqrt* and *fprem* are used.
- PORTS are implemented using the device access instructions.
- Counted protocols are implemented using the variable length I/O instructions *vin* and *vout*.
- Care has been taken in the code generation and optimization to ensure that effective use is made of the IMS T9000's workspace cache and CPU pipeline.

Access to the full IMS T9000 instruction set is available through assembly inserts.

### 3.4.3 TCOFF

The binary code produced by INMOS T9000 OCCAM 2 Toolset tools is in Transputer Common Object File Format (TCOFF). This allows integration with other TCOFF-based utilities, including the INMOS T9000 ANSI C Toolset and the INQUEST development environment.

### 3.4.4 Separate compilation

Collections of subprograms can be compiled separately using the INMOS OCCAM 2 compiler and optionally combined into a library. The linker is used to combine separately compiled functions and procedures as a linked unit. A single copy of a linked unit cannot be distributed over more than one processor. The linker supports selective loading of library units.

### 3.4.5 Mixed language programming

The INMOS T9000 ANSI C and occam 2 compilers are fully compatible and allow simple mixing of languages. occam and C processes may be freely mixed when configuring a program for a single transputer or a network of transputers. Such processes will run in parallel and communicate using channels.

The INMOS T9000 occam 2 Toolset supports calling C functions directly from occam just like other occam procedures. Pragmas are provided to tell the compiler to generate the hidden static link parameter (which is required by C but not by occam) and to change the external name of a function, since C functions may have names which are not legal occam procedure names.

Similarly, the INMOS T9000 ANSI C Toolset allows occam 2 procedures and single-valued occam 2 functions to be called from C just like other C functions.

### 3.4.6 Assembler code

The INMOS T9000 occam 2 Toolset provides a very powerful assembler insert facility. Assembler code can be written at any point in the source code to achieve direct access to transputer instructions for reasons of speed or code size. Full access is available to the IMS T9000 transputer instruction set and occam program variables.

The assembler insert facility supports:

- Access to the full instruction set of the IMS T9000 transputer
- Symbolic access to variables
- Pseudo-operations to load multi-word values into registers
- Loading results of expressions to registers using the pseudo-operations.
- Labels and jumps
- Directives for instruction sizing, stack access, return address access etc.

## 3.5 Libraries

The INMOS T9000 occam 2 Toolset provides a wide range of OCCAM libraries, including mathematical functions, string operations and input/output functions. The libraries support similar functions across the full range of transputer types, making it easy to port software between transputer types. Sources of most of the libraries are provided, for adaptation if required.

The libraries provided are listed below.

### occam compiler library

This is the basic occam run-time library. It includes: multiple length arithmetic functions; floating point functions; IEEE arithmetic functions; 2D block moves; bit manipulation; cyclic redundancy checks; code

execution; arithmetic instructions. The compiler will automatically reference these functions if they are required.

#### **snglmath.lib, dblmath.lib**

Single and double length mathematics functions (including trigonometric functions). These libraries use floating point arithmetic and will produce identical results on all processors. The OCCAM source code is also provided.

#### **string.lib**

String manipulation procedures. The OCCAM source code is also provided.

#### **hostio.lib**

Procedures to support access to the host terminal and file system through the file server. The OCCAM source code is also provided.

#### **streamio.lib**

Procedures to read and write using input and output streams. The OCCAM source code is also provided.

#### **msdos.lib**

Procedures to access certain DOS specific functions through the file server. The functions are specific to the IBM PC. The OCCAM source code is also provided.

#### **crc.lib**

Procedures to calculate the cyclic redundancy check (CRC) values of strings.

#### **convert.lib**

Procedures to convert between strings and numeric values.

#### **xlink.lib**

Procedures implementing link communications which can recover after a communication failure.

### **3.6 Configuration, initialization and loading**

Configuration is the process of defining how an application program is to be run on the available hardware. Given a description of the hardware network, the software network of an application and the mapping between them, the INMOS T9000 OCCAM 2 Toolset produces a *bootable file* which can be sent to the network for execution. The hardware network description is also used to prepare a *network initialization file* which is used to initialize the network and bootstrap the processors into a state where they are ready to receive the bootable file.

A configuration description file is used as input to a tool known as the *configurer*. The description is written in an OCCAM-like language which is upwards compatible with the configuration language in the Dx305 Professional OCCAM 2 Toolset. The configuration description describes the application as a network of processes and channels. It indicates which linked files should be used as the code for each process, and it also indicates on which processor each process is to be run.

The hardware description is separate from the rest of the configuration description. It is written in the INMOS Network Description Language (NDL). The configuration description refers to the NDL description of the hardware by means of a `#NETWORK` directive. The hardware description describes the processors and routing devices in the network and their connections.

The main features of the configuration tools are as follows:

- To initialize, boot and load an IMS T9000 network, it is not necessary to write any low-level code; instead the attributes of devices are specified by simple textual statements in the network description, and the low level code is generated automatically by the tools.
- The network description for a particular network can be written once, when the hardware is designed, and does not normally need to be changed between different applications.



- The configuration tools will check the correctness of the network description; this includes checking the labelling of the routing chips for deadlock freedom.
- Channels specified by the user in the configuration description are automatically mapped by the tools. In a system with routing devices, the configuration tools calculate the routing headers required for the user's channels. If there are no routing devices in the system, the configurator adds any software through-routing mechanisms required. Thus the software network is not constrained by the topology of the hardware network.
- For a particular application, the tools can produce either a bootable file which can be used to load the network from a host, or a boot-from-ROM file which can be used to program a ROM on one of the processors in the network.
- Initialization of individual IMS T9000 devices in the network (for example, setting up the external memory interface) can be done over the control link, or from an initialization ROM local to each IMS T9000. Support for generation of local initialization ROMs is included.

**3.6.1 Network description**

The Network Description Language (NDL) is used to describe the available hardware – the types of processors, their attributes and how they are connected. Processor attributes include, for example, a description of its memory map and its link speeds. The NDL also describes any packet routing switch devices in the network and their attributes. Attributes of a routing device include the *labelling* of the routing device, which indicates how packets from processors should be routed through it.

Figure 2.7 shows the NDL network description for the example network shown in Figure 2.10.

The network description will not normally change unless the hardware is changed. This NDL description of the system is used by the tools for a variety of purposes, including initializing the hardware and mapping application code onto processors. The tools can either read and check the NDL source directly or read a binary version produced by the NDL compiler, `ind1`.

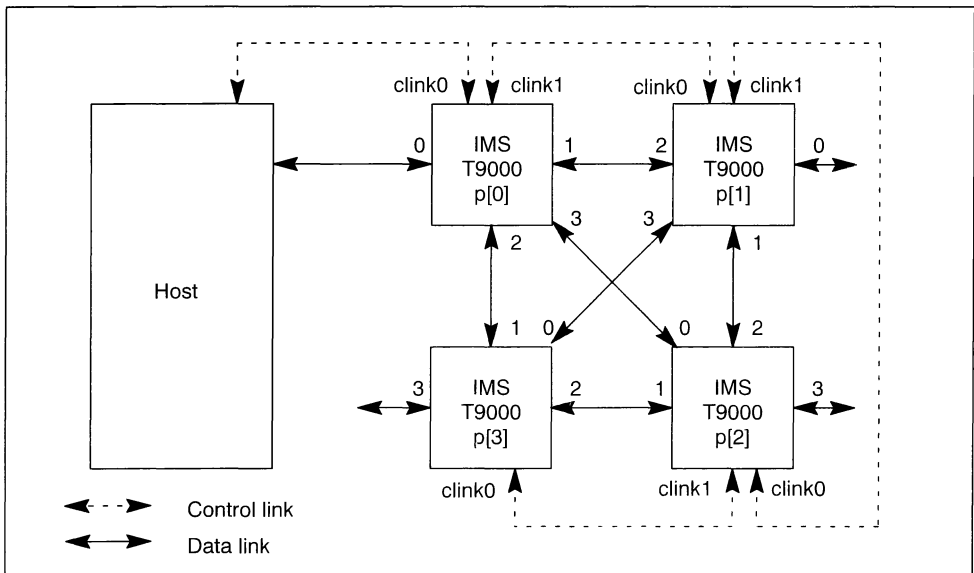


Figure 3.7 IMS T9000 network example showing link numbers

```

#include "stdndl.inc"

— Name of file with memory timing information, etc.
VAL memconfig IS "T9000.mem" :

VAL n          IS 4 :                — Number of transputers
VAL SysMem     IS 4*K :              — Size of system RAM
VAL UsrBase    IS #80000000 :        — Start address of user RAM
VAL UsrMem     IS (2*M) - SysMem :   — Size of user RAM
VAL SysBase    IS UsrBase+UsrMem :   — Start of system RAM
                                       — (= top of user RAM)
VAL memoryflags IS [[SysBase, SysMem, RAM + SYSTEM],
                   [UsrBase, UsrMem, RAM + USER]] :

[n]NODE p :                          — n transputers
CONTROLPORT host :
ARC          hostlink :

NETWORK fourT9
DO
  — set link speeds
  SET DEFAULT (link.speed.multiply := 10)
  SET DEFAULT (link.speed.divide   := [1])
  SET DEFAULT (control.speed.divide := [8])

  — set processor types
  DO i=0 FOR n
    SET p[i] (type := "T9000")
  SET p[0] (root := TRUE)
  DO i=1 FOR n-1
    SET p[i] (root := FALSE)

  — set memory information
  DO i=0 FOR n
    SET p[i] (memconfig, memory := memconfig, memoryflags)

  — connect control network
  CONNECT host[control] TO p[0][control.up]
  DO i=0 FOR n-1
    CONNECT p[i][control.down] TO p[i+1][control.up]

  — connect data links
  CONNECT host[data] TO p[0][link][0] WITH hostlink
  DO i=0 FOR n-1
    CONNECT p[i][link][1] TO p[i+1][link][2]
  CONNECT p[n-1][link][1] TO p[0][link][2]
  DO i=0 FOR n-2
    CONNECT p[i][link][3] TO p[i+2][link][0]
:

```

Figure 3.8 NDL network description example

### 3.6.2 Memory configuration

The IMS T9000 programmable memory interface supports a wide variety of memory configurations. It can be set up to provide the appropriate signals and timing needed by the memory being used. The parameters to initialize the memory interface may be sent to the transputer using the control link or may be included in the bootstrap code in ROM.

The memory configuration tool, `i.mem`, is used to assess and define memory interface parameters for each IMS T9000. It can be run interactively or in batch mode. The output may be incorporated into the network initialization file or into a ROM. The memory configuration description will not normally change unless the hardware is changed. `i.mem` can also be used to generate memory interface timing documentation.

### 3.6.3 Initializing and loading

A hosted IMS T9000 network is initialized by first sending control commands to the transputers and routers connected to the control network and then loading code using the data network.

The correct sequence of control network commands is held in a network initialization file, which contains all the information needed to initialize a system through the control links prior to loading the application code. It can be automatically generated by the initialization file generator tool, `inif`, from the network description and memory configurations. To initialize the network, the network initialization file is used by the initialization software to generate the correct sequence of commands to send to the control link network.

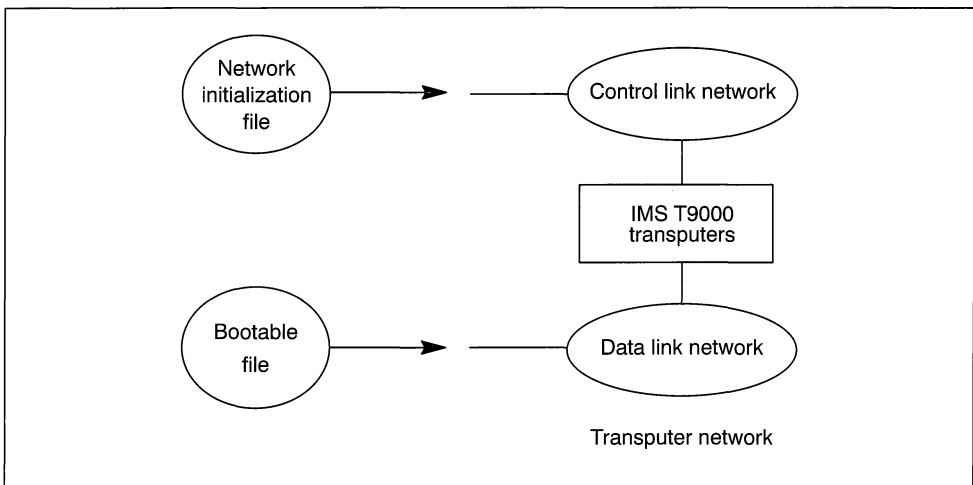


Figure 3.9 Initializing a hosted IMS T9000 network

The bootstrap code for each processor, loading code and user application code are all incorporated in the bootable file with the necessary routing information. The bootable file is generated automatically by the collector, `icollect`, from the configuration information and linked application code files. To load the code onto the network, the bootable file is sent down the data link to the data link network.

### 3.6.4 Configuration language

An OCCam-like configuration language is used to describe the network of processes and channels and the mapping of the processes onto the transputer network. The transputer network is described separately in the network description file. Multiple processes may be mapped onto the same transputer. The network description file is referred to by means of a `#NETWORK` configurer directive. This allows the user to map processes in the configuration description onto the processors named in the network description file. The routing of channels may be generated automatically or may be included in the configuration description.

The following example illustrates just how easy it is to configure a program for transputers. Instead of using the top level single transputer code shown in Figure 2.5, it may be preferred to distribute the program over a network of processors, as shown in this example.

Figure 2.9 shows the configuration text for describing the software network of processes and channels and mapping the software onto the hardware shown in Figure 2.10 and described in NDL in Figure 2.7. The `mux` and `worker` processes in the software network have been compiled and linked into the files `mux.lku` and `wkr.lku` respectively. The NDL shown in Figure 2.7 is in the file `fourT9.ndl`. The software description in the configuration text replaces the top level of code shown in Figure 2.5. In this example the `mux` process runs on the root transputer, called `p[0]`, while the individual `worker` processes run one on each of the transputers labelled `p[1]` to `p[3]` in Figure 2.10.

```
#NETWORK "fourT9.ndl"

#INCLUDE "hostio.inc"
#USE "mux.lku"
#USE "worker.lku"

CONFIG
CHAN OF SP hostin, hostout:
PLACE hostin, hostout ON hostlink:
[3]CHAN OF SP wkrIn, wkrOut:
PAR
PROCESSOR p[0]
mux(hostin, hostout, wkrOut, wkrIn)
PAR i = 0 FOR 3
PROCESSOR p[i+1]
worker(wkrIn[i], wkrOut[i])
:
```

Figure 3.10 Configuration

### 3.6.5 ROM support

The software can usually be developed and tested in a hosted development system without ROM by using host files loaded via transputer links. As a final stage, the completed application software and initialization data can be loaded into EPROMs using the EPROM program formatting tool, `ieprom`.

The EPROM formatting tool can produce a file suitable for programming a ROM; this may be either a system ROM or a local ROM. A system ROM holds network initialization data and application software for initializing and loading onto a network of transputers. Stand-alone systems will need to boot from a system ROM. A local ROM holds local initialization data for only one transputer in a network. One or more transputers in a network may be initialized from local ROMs, whether the application is loaded from a host file or from a system ROM.

The EPROM formatting tool may read the network description, memory configurations and the collected application software. From this data, the EPROM formatting tool can build a ROM file incorporating the network initialization data and application software. A control file allows the user to control how the data is arranged in the ROM. The ROM file may be produced in ASCII hexadecimal, Intel hexadecimal, extended Intel hexadecimal, Motorola S-record or binary format.

## 3.7 Connecting IMS T9000 networks to T2/T4/T8 networks

IMS T9000 transputers have improved physical links and data protocols from the T2/T4/T8 family of devices. In addition the method of initializing and controlling networks of devices is enhanced; IMS T9000 networks have a network of control links which is used to control (reset, bootstrap, monitor and analyze) devices in the network.

The IMS C100 is a protocol converter chip for connecting IMS T9000 and T2/T4/T8 networks together, and for converting the data communication and control interfaces. For detailed information on the IMS C100, please consult the appropriate data sheet.

It is envisaged that IMS T9000 users will adopt one of the following options, which are all supported by the software:

- 1 Use only T9000 family devices. Existing users of T2/T4/T8 networks can recompile existing T2/T4/T8 code onto a new IMS T9000 network.
- 2 Connect a T2/T4/T8 network to an T9000 family network, using an IMS C100 system protocol converter as a gateway between the networks, as shown in Figure 2.11. The T9000 occam 2 Toolset should be used for the IMS T9000 network and the Dx305 Professional OCCam 2 Toolset should be used for the T2/T4/T8 network. The two networks can be loaded separately or either can be loaded by the other.
- 3 Build an IMS T9000 network with T2/T4/T8 products as peripherals, using IMS C100 system protocol converters (in mode 2) as interfaces. The code for each T2/T4/T8 transputer should be compiled and linked using the Dx305 Professional OCCam 2 Toolset. The T2/T4/T8 transputers can be loaded by the IMS T9000 network.

When networks of different types are connected together (i.e. in options 2 and 3 above), the user can arrange for the networks to be loaded separately (from a host file or from ROM) or one network can load the other. The INMOS T9000 OCCam 2 Toolset includes support for initializing and loading one type of network from the other type, as follows:

- The IMS C100 system protocol converter can be used (in mode 1) to allow a T2/T4/T8 processor to control an IMS T9000 network (reset the network, initialize it and load it with an application). Software is provided in the INMOS T9000 occam 2 Toolset for inclusion in users' applications, to drive the IMS C100 from a T2/T4/T8 processor to perform these control functions.
- The IMS C100 system protocol converter can be used (in mode 2) to allow an IMS T9000 processor to control a T2/T4/T8 network (reset it and load it with an application). Software is provided in the INMOS T9000 occam 2 Toolset for inclusion in users' applications, to drive the IMS C100 from an IMS T9000 to perform these control functions.

The IMS C100 has two other modes (modes 0 and 3) which are not used by this toolset.

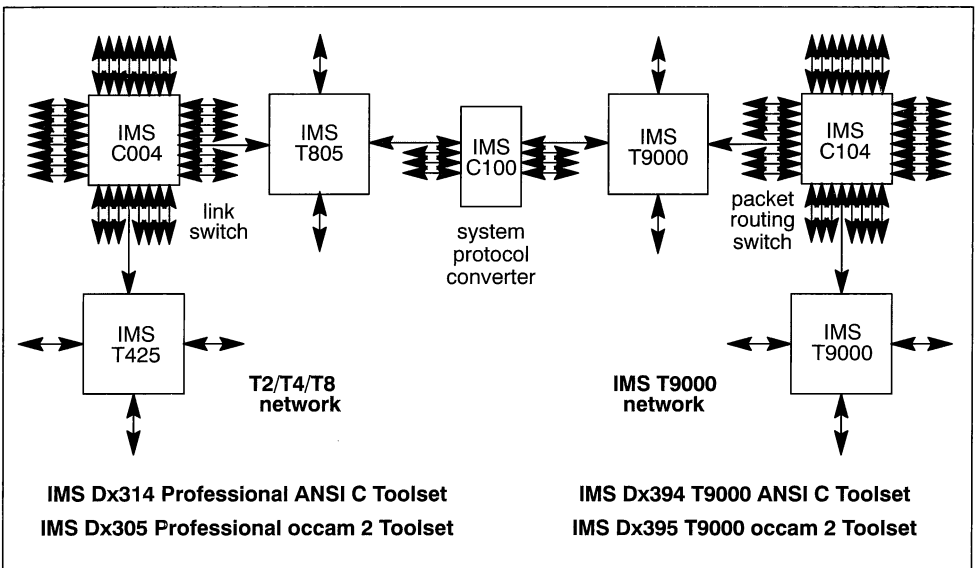


Figure 3.11 A mixed network example

## 3.8 Product components

### 3.8.1 Tools

occam 2 compiler, linker and librarian – `oc`, `ilink`, `ilibr`  
Makefile generator, binary lister program and memory map lister – `imakef`, `ilist`, `imap`  
Configuration tools – `indl`, `imem`, `inif`, `occonf`, `icollect`  
EPROM programming tool – `ieprom`  
Mixed language support library – `callc.lib`  
Mathematics functions (includes `sin`, `cos`, etc.) – `snglmath.lib`, `dblmath.lib`  
String manipulation procedures – `string.lib`  
File and terminal i/o procedures – `hostio.lib`, `streamio.lib`  
Access to certain MS-DOS calls – `msdos.lib`  
CRC calculation procedures – `crc.lib`  
String / number conversion procedures – `convert.lib`  
Extraordinary link handling procedures – `xlink.lib`  
Configuration support libraries  
Programming examples  
Makefile generator source files  
Configuration support library source files  
Network description examples  
Network control software source files  
occam libraries source files

### 3.8.2 Documentation

Toolset User Guide  
Tools Reference Manual  
Libraries Reference Manual  
System Configuration Guide  
Delivery Manual  
occam 2 Toolset Handbook  
occam 2 Language Reference Manual  
Tutorial Introduction to OCCAM

## 3.9 Product variants

### 3.9.1 Sun-4 product

#### Product

- IMS D4395 T9000 occam 2 Toolset

### Operating requirements

For Sun-4 hosted cross-development the following will be required:

- A Sun-4 workstation or server
- SunOS 4.1.1 or later
- 10 Mbytes of free disk space.

For loading target systems, a suitable transputer network interface will be required.

### Distribution media

Sun-4 software is distributed on DC600A data cartridges 60 Mbytes, QIC-24, tar format.

### Licensing

The IMS D4395 T9000 Occam 2 Toolset is a four-user product. For each product purchased, up to four users are able to use the Toolset concurrently at any one customer site. The tools can be run on any Sun-4 machine that is part of a network connected to a single machine where the licence manager is installed. Further information about the licence manager is included in the product Delivery Manual. Multiple copies can be purchased for larger project teams using volume discount curves.

No licence fee is charged for including INMOS libraries in customer products when linked with customer applications using the INMOS linker, `ilink`. Example programs and other sources provided may be included in software products, but INMOS Limited retain original copyright. Full licensing details are available from SGS-THOMSON Sales Offices, Regional Technology Centers and authorized distributors.

## 3.9.2 IBM product

### Product

- IMS D7395 T9000 Occam 2 Toolset

### Operating requirements

For PC hosted cross-development one of the following will be required:

- IBM PC with a 386 or 486 processor and a minimum of 4 Mbytes memory

In each case the following will be required:

- DOS 5.0 or later
- 9 Mbytes of free disk space

For loading target systems, a suitable transputer network interface will be required.

### Distribution media

Software is distributed on two media systems, both of which are supplied in the product:

- 1.2 Mbytes (96TPI) 5.25 inch IBM format floppy disks
- 1.44 Mbytes 3.5 inch IBM format floppy disks.

### Licensing

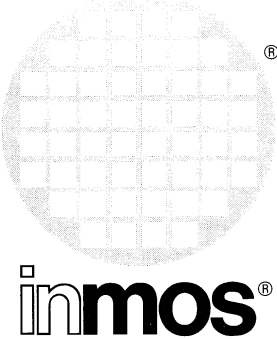
The IMS D7395 T9000 Occam 2 Toolset is a single-user product. Multiple copies can be purchased for larger project teams using volume discount curves.

No licence fee is charged for including INMOS libraries in customer products when linked with customer applications using the INMOS linker, `ilink`. Example programs and other sources provided may be included in software products, but INMOS Limited retain original copyright. Full licensing details are available from SGS-THOMSON Sales Offices, Regional Technology Centers and authorized distributors.

### **3.10 Problem reporting and field support**

A registration form is provided with each product. Return of the registration form will ensure you are eligible for future product updates. Software problem report forms are included with the software. INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centers and authorized distributors.

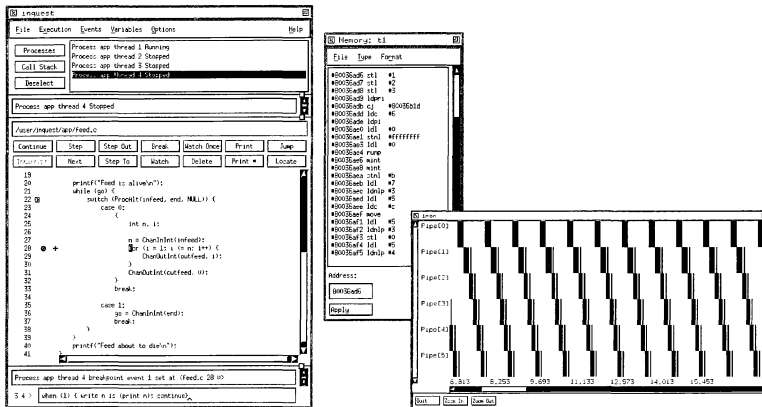




# IMS D4390 T9000 INQUEST

## Preliminary Information

The information in this datasheet is subject to change.



### KEY FEATURES

- Supports INMOS T9000 ANSI C and T9000 occam 2 Toolsets.
- Supports IMS T9000 transputer.

### Interactive hosted windowing debugger:—

- Windowing interface using X Window System and OSF/Motif.
- Programmable command language.
- Process and thread break points.
- Single stepping of threads.
- Read/Write/Access watch point capability.
- Source code or assembly code view of a thread.
- Stack trace-back facility.
- Variable and memory display facility.
- Expression interpreter.
- Facilities to interrupt and find threads.

### Performance analysis tools:—

- Analysis of time spent in each function.
- Analysis of processor idle time and high priority time.
- Analysis of processor load over time.

### DESCRIPTION

The *IMS D4390 T9000 INQUEST* development environment package provides three additional tools for developers using the INMOS T9000 ANSI C Toolset or INMOS T9000 occam 2 Toolset for IMS T9000 transputer networks hosted by Sun-4 workstations. An *interactive windowing debugger* provides single stepping, breakpoints, watchpoints and many other features for debugging sequential and parallel programs running on single or multiple transputer networks. An *execution profiler* gives a post-mortem statistical analysis of the total execution time used by each function in a program. A *transputer network utilization monitor* shows graphically when each processor in a network was busy and when it was idle.

## 4.1 Product overview

This document contains advance information for the *IMS D4390 T9000 INQUEST* development environment.

The INMOS T9000 INQUEST development environment provides three additional tools for developers using the IMS D4394 T9000 ANSI C Toolset or IMS D4395 T9000 OCCAM 2 Toolset for IMS T9000 transputer networks hosted by Sun-4 workstations.

The debugger supplied in the INQUEST package supports source-level and low-level debugging of multi-process and multi-processor programs. A graphical user interface, with multiple windows, simplifies use of this sophisticated tool. The debugger provides single stepping, breakpoints, watchpoints and many other features for debugging sequential and parallel programs running on single or multiple IMS T9000 networks.

An execution profiler gives a post-mortem statistical analysis of the total execution time used by each function in a program. A transputer network utilization monitor shows graphically when each IMS T9000 processor in a network was busy and when it was idle.

The IMS D4390 T9000 INQUEST development environment operates in a host-target transputer development environment where the target network is accessed using the IMS B300 Ethernet to Transputer Gateway product. The debugger runs on the host computer from which the IMS T9000 network has been loaded. Support is also included for installation of other host / transputer interfaces.

The target network running the IMS T9000 transputer program may be made up of custom-built boards or INMOS T9000 board products. A suitable interface board from IMS T2xx/T4xx/T8xx to IMS T9000 can be used to provide the bridge from the OS-Links of the IMS B300 Ethernet to Transputer Gateway to the DS-Links of the IMS T9000 network.

### 4.1.1 Applications

- Single- and multi-processor embedded systems.
- Massively parallel applications.
- Evaluation of concurrent algorithms.
- Scientific programming.

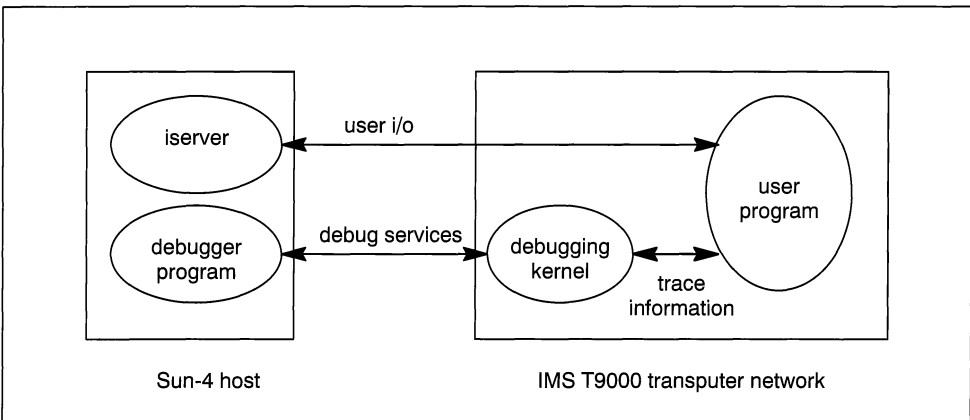


Figure 4.1 Debugger architecture

## 4.2 Interactive windowing debugger

The interactive debugger consists of a host-based symbolic debugger and transputer-resident distributed debugging kernels that are configured into the application program on one or more processors.

A user interface, built using the X Window System and OSF/Motif, displays source code or disassembled code and enables the user to interact with the debugger by means of buttons and menus, mouse and keyboard.

The program being debugged may consist of any number of parallel threads of execution, some of which may be running while others are stopped or being single stepped. The host debugger program is asynchronous and holds a copy of the last stopped state of each thread, so values may be inspected by the host while the user program is running on the transputer network. Multiple debugging windows may be opened to view different parts of the program simultaneously.

The interactive debugger provides the following features:–

- A break point facility that can be used on particular threads of execution.
- A single stepping facility that allows a thread of execution to be single stepped at the source level or at the assembly code level.
- A watch point facility that enables the program to be stopped when variables are to be written to or read from.
- A facility to find the threads of execution of a program and set break points on them.
- A stack trace facility.
- A facility to monitor the creation of threads of execution.
- Commands to print the values of variables and display memory.
- A simple interpreter to enable C and OCCAM structured variables to be displayed.
- A programmable command language that allows complex break points and watch points to be set and enables debugging scripts to be generated.
- A source and object browser to select a process, a thread and source code.

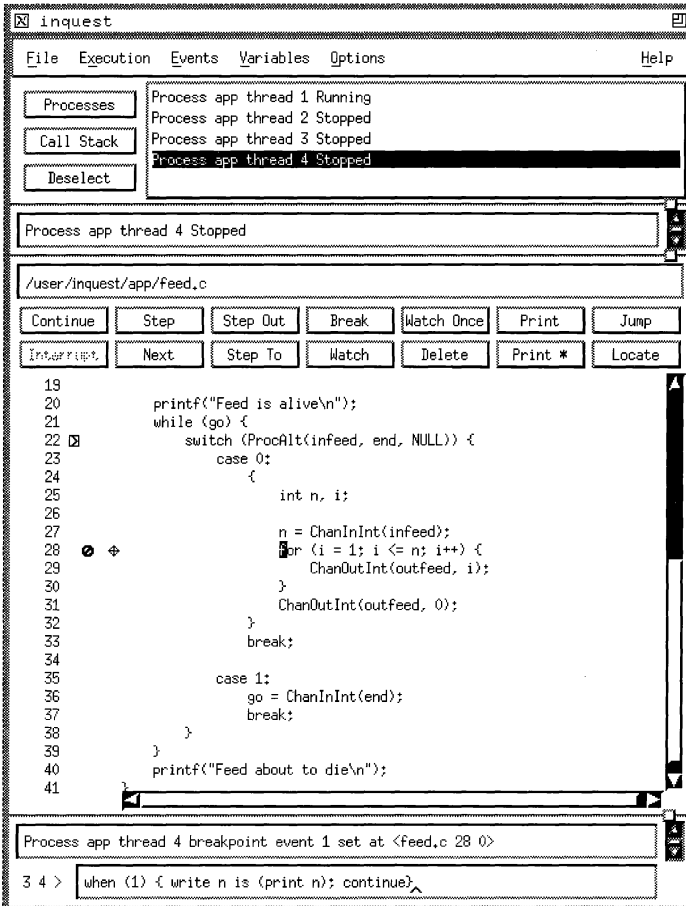


Figure 4.2 Selecting a process

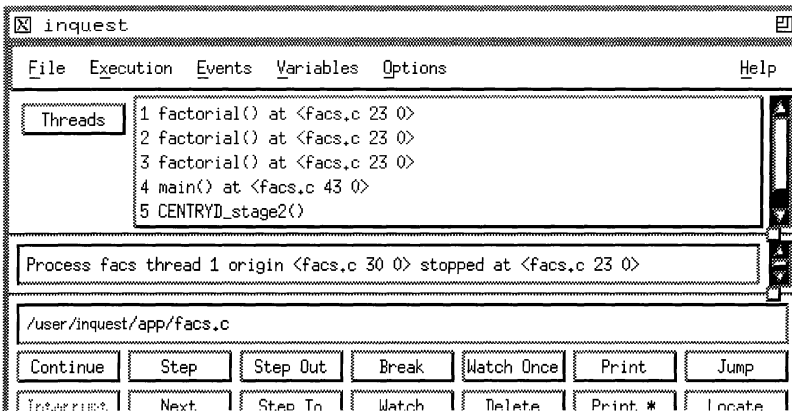


Figure 4.3 A stack trace

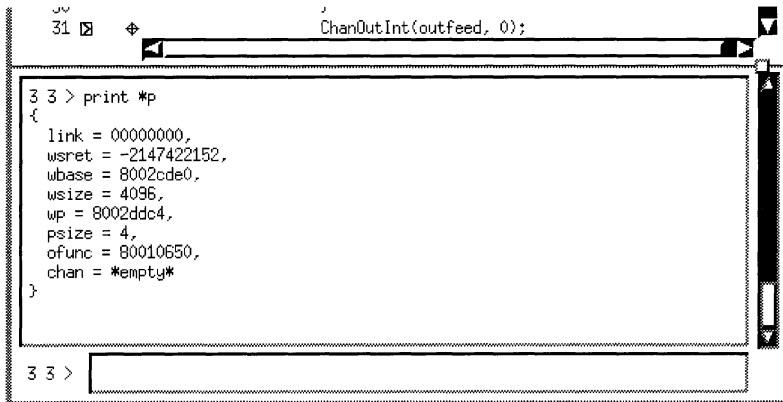


Figure 4.4 Displaying a structured variable

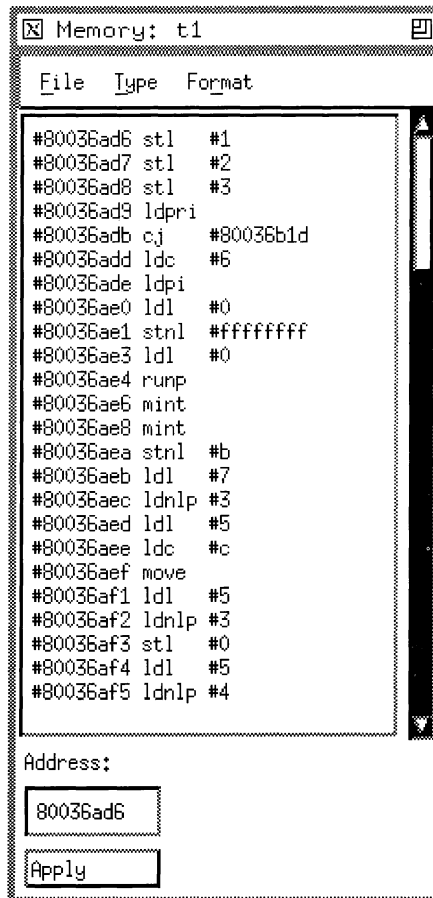


Figure 4.5 Displaying memory contents

### 4.3 Performance analysis tools

The performance analysis tools comprise an execution profiler and a network utilization monitor. With these tools the execution of the user program is monitored by a distributed profiling kernel. The monitoring data is stored locally to each processor, so the profiling tools have little execution overhead on the application. After the program has completed execution, the monitoring data is extracted from the processors and is analyzed to provide displays on the program execution.

#### 4.3.1 Execution profiler

The execution profiler gives an analysis of the total time spent executing each function on each processor. It provides the following information on program execution:–

- The percentage time spent executing each low priority function.
- The percentage time spent executing at high priority.
- The percentage idle time of each processor in the network.
- The percentage time spent executing each configuration process.

```

Processor "Root"
Idle time 35.3% (19516)
High time 0.1% (37)
Wptr Misses 0
Iptr Misses 0
Resolution 4

-----
Process "ex" (99.9% processor) (35.666s)
Stack 100.0% (35666)   Heap 0.0% (0)   Static 0.0% (0)
Function Name          | Process | Processor | Samples
-----
libc.lib/getc         | 11.4    | 11.4      | 4081
cc/pp.c/pp_rdch0     | 10.1    | 10.1      | 3605
cc/bind.c/globalize_memo | 6.9     | 6.9       | 2467
cc/pp.c/pp_process   | 4.3     | 4.3       | 1525
cc/pp.c/pp_rdch3     | 4.2     | 4.2       | 1497
cc/pp.c/pp_rdch2     | 3.9     | 3.9       | 1380
cc/pp.c/pp_rdch1     | 3.8     | 3.8       | 1354
cc/pp.c/pp_rdch      | 3.5     | 3.5       | 1252
cc/pp.c/pp_nextchar  | 3.3     | 3.3       | 1189
cc/pp.c/pp_checkid   | 3.2     | 3.2       | 1150
cc/lex.c/next_basic_sym | 2.7     | 2.7       | 979
libc.lib/strcmp      | 2.3     | 2.3       | 812
libc.lib/DummySemWait | 2.2     | 2.2       | 784
libc.lib/sub_vfprintf | 1.7     | 1.7       | 617

```

Figure 4.6 Example output from the execution profiler

#### 4.3.2 Utilization monitor

The utilization monitor shows in graphical form the utilization of each processor over the time of the program execution. This is displayed in the form of an interactive program using X Window Systems and OSF/Motif that displays a chart of processor execution against time.

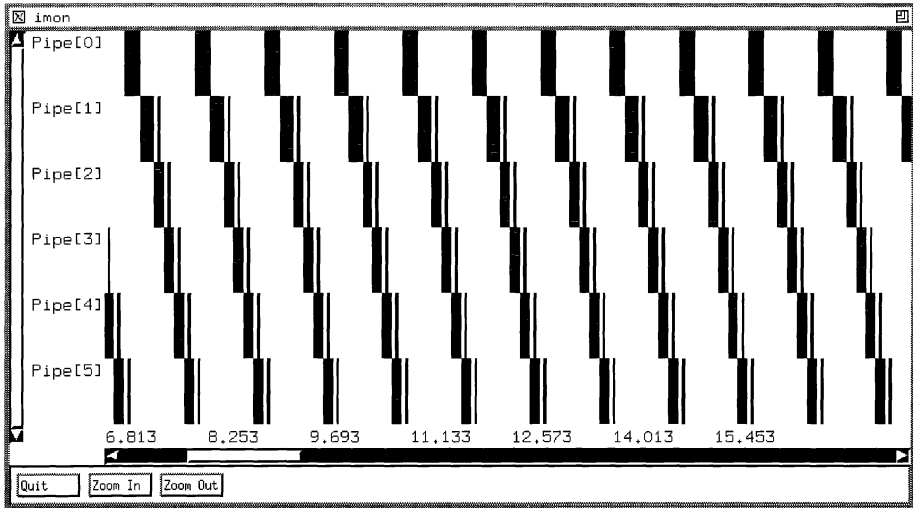


Figure 4.7 Example display from the utilization monitor

## 4.4 Product details

### 4.4.1 Operating requirements

The IMS D4390 T9000 INQUEST package will run on a Sun-4 SPARC-based workstation, server or compatible, running SunOS 4.1.1 or later. To support the graphical user interface, the environment must include an X Window System server (X11R4 or later) such as an X terminal or SPARCstation running OpenWindows 3.

The IMS D4390 product is designed to operate in conjunction with the IMS D4394 T9000 ANSI C Toolset or IMS D4395 T9000 OCCam 2 Toolset software and the IMS B300 Ethernet to Transputer Gateway.

The IMS D4390 software is run in conjunction with a 'floating' licence manager. For each product purchased, up to four users are able to use the package concurrently at any one customer site. The tools can be run on any Sun-4 machine that is part of a network connected to a single machine where the licence manager is installed. Further information about the licence manager is included in the product *Delivery Manual*.

### 4.4.2 Documentation

The INMOS T9000 INQUEST package comes complete with a full documentation set and tutorial guide. Example applications used in the tutorial and documentation are included in source form on the distribution media.

### 4.4.3 Distribution media

Sun-4 software is distributed on DC300A data cartridges 60 Mbyte, QIC-24, in tar format.

## 4.5 Problem reporting and field support

INMOS TRAMs and transputer toolkit development products are supported world-wide through SGS-THOMSON Sales Offices, Regional Technology Centers, and authorized distributors.







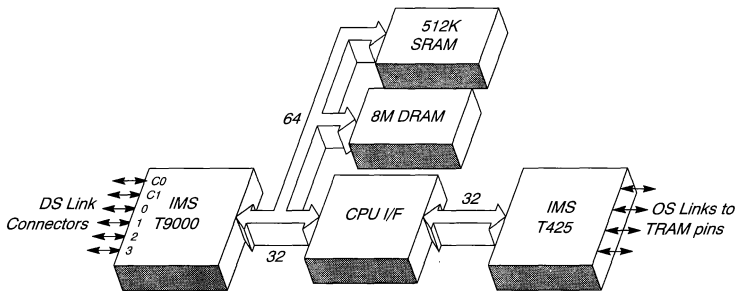
**inmos**<sup>®</sup>

# IMS B490

## IMS T9000 Development TRAM

Preliminary information

The information in this datasheet is subject to change



### FEATURES

- IMS T9000 processor provides 200MIPS and 25MFLOPS (50MHz processor).
- Three 64 bit banks of memory, two 4Mbyte banks of fast, page mode, DRAM and one 512Kbyte bank of two cycle access SRAM.
- All IMS T9000 links buffered and available on DS link modular connectors. Four DS data links provide 100Mbits/s communication with other T9000s.
- Support software allows use with T9000 toolsets. Applications can be run on the IMS T9000 on the IMS B490 or on a T9000 network connected to the IMS B490.

### DESCRIPTION

The IMS B490 is the first *iq* systems board to support the IMS T9000. It is designed to be used as an evaluation board for the IMS T9000; and as a development board allowing control of T9000 networks from a TRAM environment.

Software supplied with the TRAM supports three modes of operation when used with the INMOS T9000 toolsets. Mode one allows applications to be loaded, run, and debugged on the T9000 via the standard TRAM interface. Mode two provides the ability to control a T9000 network via a control and data DS-Link pair connected to two of the data links of the on board T9000. The final mode allows IMS B490 to be used as a T9000 in a network providing a means of constructing simple T9000 networks from IMS B490.

## 5.1 Introduction

The IMS B490 allows the evaluation and development of IMS T9000 systems while working within a TRAM based environment.

The IMS T9000 is the latest member of the transputer family and is designed for embedded systems and high performance computing applications. It integrates a 32 bit integer processor, a 64 bit floating point processor (IEEE 754 compliant), 16Kbytes of cache memory, a communications processor (VCP), and four high bandwidth serial communications links.

Key features of the IMS T9000:

- Pipelined super-scalar architecture.
- 100Mbits/s serial data links.
- Multiple virtual channels on data links between devices.
- 16Kbyte on-chip cache.
- 64-bit wide memory interface.
- Error trapping and handling.
- Separate control and data link networks.

Full information on the IMS T9000 can be found in [1].

The IMS B490 hardware conforms to the current TRAM standard so that the wide range of existing motherboards and host interfaces such as the IMS B008 and the IMS B300 can be used while developing T9000 based systems. In conjunction with the IMS B490 support software and a T9000 toolset these hardware and software components provide the tools to develop and evaluate T9000 systems. In the future *iq* systems will be supporting T9000 development and systems building with a range of hardware and software components based around the HTRAM standard and DS-Links. For further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs see references [2] and [3] which are included in *The Transputer Development and iq systems Databook* [4].

Full information on the *HTRAM Specification* can be found in Appendix A.

## 5.2 IMS B490 operation

The IMS B490 TRAM incorporates both a T425 and a T9000 transputer. The T425 transputer is connected to the OS<sup>1</sup> link pins on the TRAM, and to the **Reset**, **Analyse**, and the **notError** pins in a similar manner to all TRAMs.

Communication between the T425 and the T9000 takes place over a FIFO mapped into the address space of both processors.

The T9000 is reset by the TRAM's **Reset** signal. The T9000 is connected to a ROM containing code to configure its memory interface and to support communication with the T425.

The T9000's control and data DS-Links are buffered and taken to connectors on the board. Cables can be used to interconnect the IMS B490 to similar standard connectors on other boards supplied by *iq* systems such as the IMS B100 HTRAM motherboard.

1. Over Sampled links are the standard for T2, T4, and T8 series processors

The IMS B490 TRAM is designed to be used for three distinct purposes:

- A single-processor T9000 program can be loaded onto the IMS B490 from a host computer, and run on the T9000, obtaining its I/O services from the host. This is primarily intended to support benchmarking of programs running on a T9000.
- The IMS B490 can be used as a controlling node for a T9000 network. In this case the IMS B490 is connected to a T9000 network by two DS-links (one for control and one for data), and controlling software for the network is run on the IMS B490. The IMS B490 does not run any part of the user application.
- The T9000 on the IMS B490 can be used as a component in a T9000 network. The T9000 is externally controlled by means of its control links and communicates externally by means of its data links. The T425 and its interface to the T9000 are not used in this mode.

These different intended uses are supported by three *modes* of operation of the IMS B490. Two jumpers are provided to set the operating mode. Note that the setting of the jumpers should not be changed while running an application on the IMS B490 as this could lead to incorrect operation.

### 5.3 Support software

The support software supplied with the IMS B490 has the following key features:

- Provides the ability to load and run programs on T9000 networks from a host computer.
- Host-network communication occurs through standard OS link to host interfaces.
- Uses T9000 initialization and bootable files produced by the INMOS T9000 toolset products.
- Supports loading and running single-processor T9000 programs on the IMS B490.
- Supports use of the IMS B490 as a controlling node for a T9000 network.
- Network Description data provided for IMS B490 so that its T9000 can be used as a node within T9000 network.
- Provides host services (filing system, user I/O) to T9000 applications.
- Supports use of interactive debugger INQUEST with programs running on IMS B490 or on connected T9000 network.

The following three sections describe the three operating modes of the IMS B490 hardware and software in more detail, including information on how to program and use the IMS B490 in each mode and the software implications.

The preparation of programs to run on T9000s requires a T9000 toolset product (ANSI C or occam 2). The IMS B490 software support product complements the toolset products, by making it possible to load and run a program from a host computer. Further details of the toolsets can be found in chapters 2, 3 and 4.

#### 5.3.1 Mode 1 : Single processor T9000 for benchmarking and evaluation

Using a IMS B490 in Mode 1, a user can run code on the single on board IMS T9000 transputer. This allows benchmarking of the IMS T9000 and, for instance, investigation of the performance of applications using the on chip cache, SRAM, and DRAM memory banks in different ways.

The IMS B490 needs to be plugged into an existing TRAM motherboard and one of its links must be connected to a link interface to a host. The host must also be able to control the **Reset**, **Analyse**, and monitor the **notError** pins of the IMS B490. Typical examples of host interface/motherboard combinations are the IMS B008 PC TRAM motherboard, or the IMS B300 ethernet interface in conjunction with TRAM motherboard such as the IMS B014.

The IMS B490 TRAM should normally be directly connected to the host (and so occupy the root position in the TRAM network). If this is so, direct communication can take place from the host to the T425 on the

IMS B490. However, if this is not possible, `iskip` can be used to skip over any intermediary T2/T4/T8 processors.

The arrangement is shown in a block diagram form below.

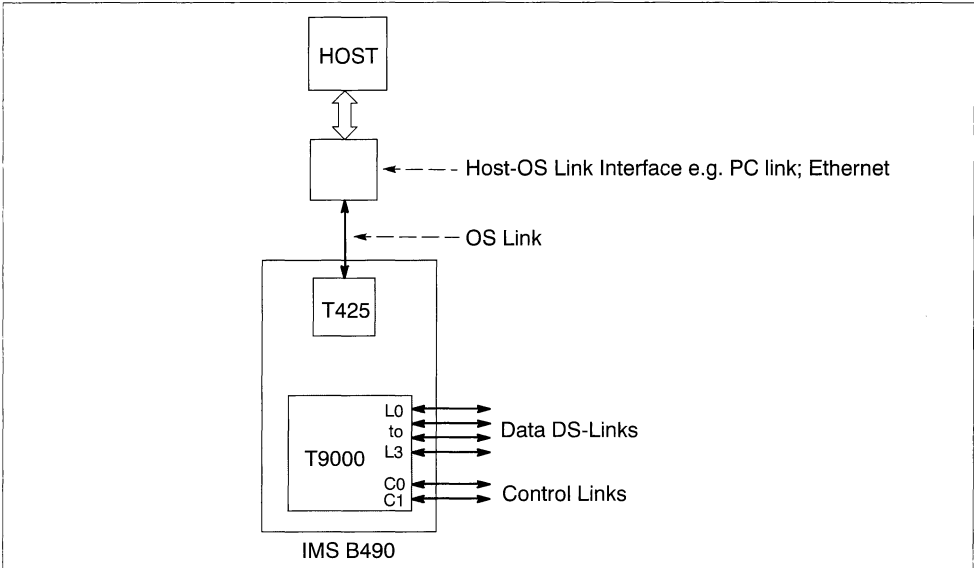


Figure 5.1 IMS B490 as a single T9000 development system

In this mode the T9000 boots from ROM, configures its memory interface, and then waits for a program to be supplied to it over the interface from the T425. The T425 boots from link, and is supplied with software from one of its OS links. This software allows the T425 to “relay” communications backwards and forwards between its boot link and the T425’s interface to the T9000.

### Programming an IMS B490 in mode 1

A program to run on a IMS B490 in mode 1 is a single-processor program prepared from a single-processor configuration description (either C-style or occam-style), and a number of C or occam program source files.

The configuration description file should refer to a Network Description Language (NDL) file. The NDL file will provide the configurer with the information it needs about the T9000 in the IMS B490 (for example, memory regions and their size). An example NDL file for the IMS B490 will be provided by INMOS.

The software description in the configuration file should describe the processes to be run on the IMS B490, the linked unit files containing their code, and the channel connections between them. A pair of channels must be placed at specific addresses in the memory, in order to obtain services from the host, via the T425.

To prepare a program for running on the IMS B490 in this mode, the following steps would be followed:

#### Compilation and linking

Compile and link the program source using the appropriate T9000 toolset compiler and linker. This will result in one or more linked unit (`.1ku`) files.

#### Configuration

Prepare a suitable configuration source file and run either the C-style configurer `icconf` or the occam-style configurer `occonf`.

#### Creation of bootable file

Run the code collector tool `icollect` on the configurer’s output; this tool then produces a bootable file.

No initialization file needs to be prepared to use the IMS B490 in this mode; all initialization is done by the T9000's ROM.

Once the bootable file has been prepared, it can be run on the T9000 in the IMS B490. A **run** command is provided to do this.

To run a bootable file on the IMS B490, invoke the **run** command, giving it the name of the bootable file. This will cause the T9000 in the IMS B490 to run the program. The program may make use of host services over the channels that were placed within the memory. Communications on these channels are routed via the T425 and appear as communications on the link connecting the host to the IMS B490.

The **run** command also starts up a host server to handle server requests coming from the application running on the T9000.

A mode 1 IMS B490 program can also be prepared for use with the INQUEST interactive debugger, by selecting the appropriate compilation and configuration options. The resulting bootable file, when run using the **run** command, will cause an interactive debugger to be started on the host for debugging the application. This requires the T9000 INQUEST development environment product.

If an error occurs within a program running on the T9000, it will communicate this to the T425, which can set the error signal on the TRAM.

### 5.3.2 Mode 2 : T9000 network controller

In this mode the IMS B490 is dedicated to controlling an attached T9000 network. The network may be a user's target system, it may be constructed from other IMS B490s (in mode 3), or from HTRAM boards on an appropriate motherboard (such as the IMS B100 HTRAM Motherboard).

Two DS-Links are taken from the IMS B490 to the attached T9000 network; one is used to control the network (the control link), and another one used as a data link over which code is loaded and communications between the network program and the host are carried out. The following diagram shows the use of the IMS B490 in this mode:

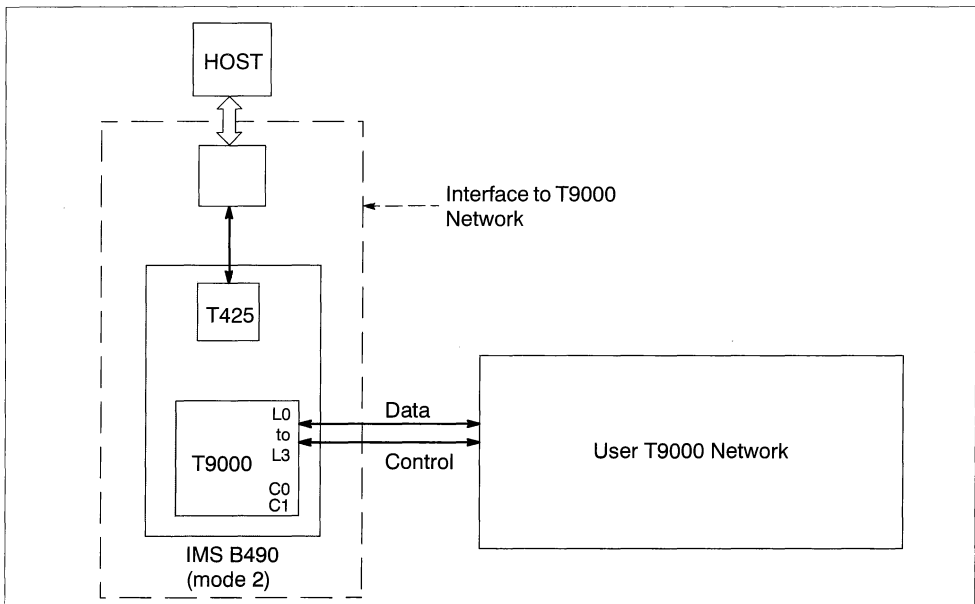


Figure 5.2 Connecting an external T9000 network via the IMS B490

In this mode the behavior of the hardware is the same as in the previous mode. However, in this case the program that the T9000 runs is not a user program, but a network controlling and loading program provided by INMOS. After initializing and loading the target network, the program running on the IMS B490 will allow the network program to use host services by passing server communications backwards and forwards between the host and the target network.

**Note:** An IMS B490 used in this mode is not able to run any part of the user application.

### Programming a IMS B490 in mode 2

In this mode the IMS B490 is being used as a network controller for a T9000 network. It does not run any part of the application, but it initializes and loads the network, relays server communications to and from the host, and supports debugging of the program running on the network.

An initialization file and bootable file are prepared for the network using the appropriate T9000 toolset. This involves the following steps:

- Writing a description of the network in the Network Description Language (NDL).
- Creating a network initialization file from the NDL description.
- Compiling and linking the program source files.
- Writing a configuration description mapping software processes to processors in the network.
- Creating a bootable file from the configuration description.

These are the steps that will always be followed when preparing a program to run on a T9000 network, and they are described in more detail in the data sheet on the appropriate toolset. The IMS B490 support software provides the ability to get the program running on a T9000 network attached to the host via a IMS B490.

A **run** command is provided to initialize the network and then load and run the program on the T9000 network.

The **run** command takes as its parameters the names of the network initialization file and the application bootable file produced by the tools. The command causes the IMS B490 to initialize the attached network, over the control link, using the information in the initialization file, and then load the application bootable over the data link.

Once the application is running in the network, the IMS B490 relays server communications between the network and the host. It also monitors the network for the occurrence of an error. During this period, the user will interact with the application in the normal way, by means of a server started in the host by the **run** command.

If an error occurs in the network, the **run** software can inform the user that an error has occurred, allowing the user to debug the network or reload the application.

The T9000 network program can also be prepared for use with the INQUEST interactive debugger, by selecting the appropriate compilation and configuration options. The resulting bootable file, when run using the **run** command, will cause an interactive debugger to be started on the host for debugging the application. This requires the T9000 INQUEST development environment product.

#### 5.3.3 Mode 3 : T9000 network node

In mode 3, the T9000 on the IMS B490 is used as a node in a T9000 network. The T425 and its interface to the T9000 are not used in this mode.

A small T9000 network can be constructed from a number of IMS B490, operating in mode 3, mounted on TRAM motherboards. The IMS T9000 on each of these IMS B490 is controlled via its control (**Control**) link, and communicates externally over its data links.

The IMS B490s in mode 3 are interconnected into networks using the DS-Link cables provided with the TRAMS. These are plugged into the data link connectors to form a data network. Further cables plugged into the control links are used to form a control pipeline.

An IMS B490 in mode 2 can be used to interface this network's control link and a data link to a server program running on a host.

This is shown in the following diagram:

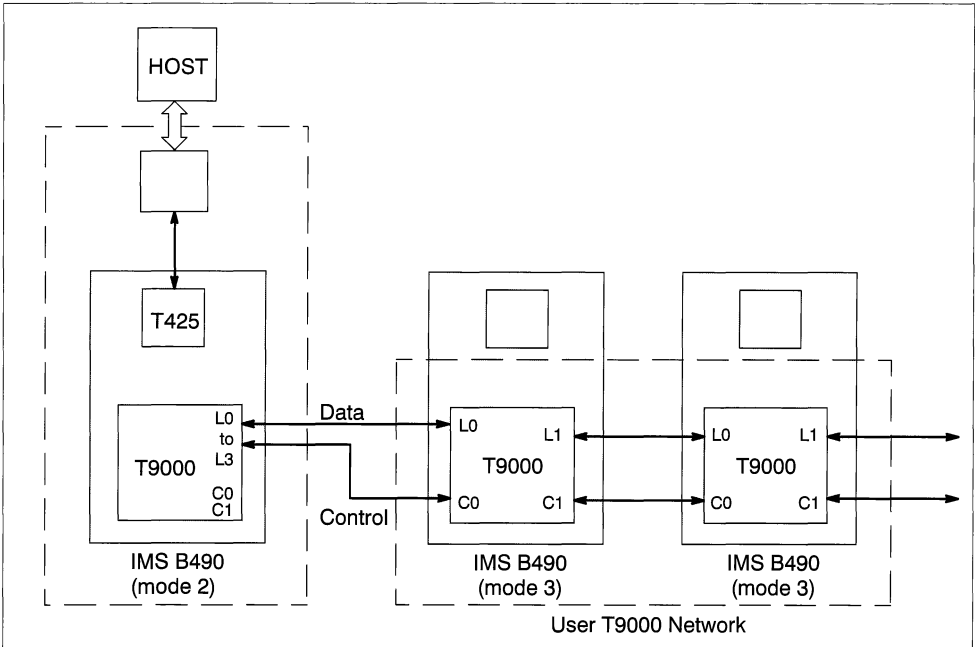


Figure 5.3 T9000 Network based on IMS B490 units

The T9000 is set so that it initially boots from link; that is, it waits for commands to arrive over its control link. Controlling software will cause the T9000 to be re-booted from its ROM, when it configures its memory interface. It then signals over its control link that this has been completed, allowing control software to do further configuration, initialization, and loading of bootstrap code.

**Programming an IMS B490 in mode 3**

In this mode the T9000 on the IMS B490 is being used as a component in a T9000 network. The configuration of the memory interface of the T9000 is carried out by firmware in the ROM under control of a controlling process via the control link as described above.

As with any T9000 network, a network constructed of IMS B490 TRAMs must be described in the Network Description Language (NDL). Using this description the tools can prepare initialization files and application bootable files for the network.

The NDL description of the IMS B490 when used in this mode is supplied in the support software package. This can then be duplicated and incorporated into a full network description. Example NDL descriptions of IMS B490 based networks are also provided.

### 5.3.4 Software component summary

This section summarizes the components of the IMS B490 support software.

#### Documentation

##### Delivery manual

A delivery manual is provided, which describes how to install and set up the IMS B490 support software.

##### User manual

A user manual is provided, which describes how to use the IMS B490 support software. This includes:

- How to use the toolset to prepare programs to run on the IMS B490.
- How to run programs on the IMS B490.
- How to run programs on a target network, using the IMS B490 as a network controller.
- Worked examples

#### Software

The following software components are supplied:

- NDL description of the IMS B490 for use in mode 1 and mode 3.
- Memory configuration description file.
- The bootable file to run on the T425.
- The network controlling software to run on the IMS B490 in mode 2.
- The `run` command software for the host.
- INMOS server software for the host.
- Example NDL and program files.

### 5.3.5 Product variants

Variants of the software are initially available for the IBM PC and Sun 4. Please contact INMOS for availability information on variants for other hosts. Both variants are shipped in the IMS B490 product.

## 5.4 Hardware description

The IMS B490 hardware consists of three major sections:

- IMS T9000 and memory banks.
- IMS T425 and memory.
- The interface between the IMS T9000 and the IMS T425.

Design of the IMS T9000 memory system on the IMS B490 has been optimized to allow the maximum performance of the T9000 to be evaluated while at the same time providing a reasonable size memory area in which to run an application. All six links, two control and four data, of the IMS T9000 are differentially buffered and brought out to connectors allowing IMS B490s to be used to evaluate DS-Links, and to control and be used in T9000 networks. The differential electrical standard used on the DS-Links of the IMS B490 will be common to all *iq* systems external DS-Link electrical connections.



The hardware on the T425 side of the IMS B490 looks very much like a standard TRAM with the addition of the interfaces necessary to allow communication with and control of the IMS T9000.

A bi-directional FIFO based interface is provided between the T425 and T9000 processors on the board. A block diagram of the IMS B490 is shown in Figure 5.4.

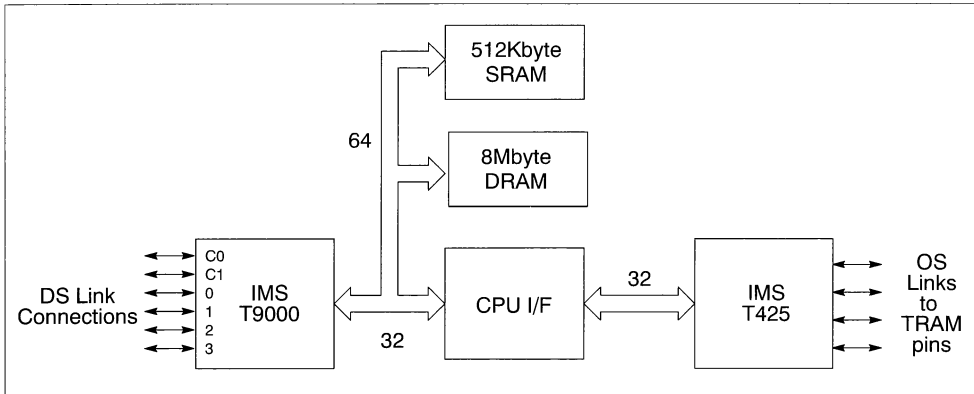


Figure 5.4 Block diagram of IMS B490 hardware

The following two sections give sufficient information on the hardware to allow users to operate the IMS B490 in its three modes in conjunction with the supplied support software, a T9000 toolset, and a user’s application. Use of the IMS T425 processor, the connected memory, and the T9000/T425 interface directly by user applications is not supported. Brief information on these parts of the IMS B490 hardware is given only to allow user system diagnosis or the running of generic software such as **iskip**. Users should avoid running software on the IMS B490 that attempts memory “sizing” operations or any accesses to the interface registers as this could lead to unpredictable operation or misleading results.

**5.4.1 IMS T425 hardware**

The IMS B490 is designed to behave like a standard TRAM from the point of view of system services (**Reset, Analyse, notError, ClockIn**) and links. The IMS T425’s systems services and link signals are connected to these signals on the TRAM pins in the same way as on most other TRAMS.

1Mbyte of DRAM provides code and data memory for the “relay” process and the T9000/T425 interface processes.

The T9000/T425 interface device registers are memory mapped into the T425’s address space. Other registers are provided to control the generation of events to the T425 in response to changes in the state of the interface and allow control of the T9000 from software running on the IMS T425. These registers and the FIFO interface are used in modes 1 and 2 to allow the loading, running, and communication with software running on the on board IMS T9000.

A memory map of the address space of the IMS T425 is given in table 3.1.

Memory/register	Hardware byte address
T425 Internal memory	#80000000 to #80000FFF
External DRAM, 1MByte	#80001000 to #80100FFF
Repeated external DRAM	#80101000 to #8FFFFFFF
FIFO data, control, event, and T9000 control registers	#00000000 to #7FFFFFFF

Table 3.1 IMS B490 T425 memory map

### 5.4.2 T9000 hardware

The T9000 hardware on the IMS B490 consists of a ROM to provide firmware for the three operating modes, a 512Kbyte 64-bit bank of fast SRAM, and two 4Mbyte 64-bit banks of DRAM. Details of the memory can be found in the memory configuration section below.

A similar set of registers to those on the T425 side of the T425/T9000 interface are memory mapped into the T9000's address space. Events are generated on the **EventIn0** input.

As with the T425 side of the interface, user applications making direct access to the T425/T9000 interface or using the Event channel are not supported and may result in unpredictable behavior.

#### Memory configuration

The Programmable Memory Interface (PMI) of the IMS T9000 provides a high performance interface to memory and I/O devices with a minimum of additional logic. The interface supports four banks of memory with the timing of strobos, length of cycles, and the basic data width plus other parameters programmable for each bank. In addition a fixed timing, 8 bit, boot bank is provided to support a separate boot ROM.

On the IMS B490 banks 1 and 2 are used to support the two banks of DRAM, bank 0 for the SRAM, and the interface registers mapped into bank 3. The firmware ROM is connected to the boot bank.

All of the memory banks, except the ROM bank, are configured to be 64 bits wide and are cacheable for maximum performance. The SRAM bank consists of 15ns access time SRAMs and is expected to provide two cycle access. Two cycle access provides the highest performance memory bandwidth the PMI supports, up to 200Mbytes/s for a 50 MHz T9000. The bank into which the interface registers are mapped is configured to be 32 bits wide with registers of less than 32 bits mapped into the least significant end of the word. This bank is not cacheable.

A memory map is given in table 3.2. The memory map is set by configuring the PMI. This is carried out by the firmware in the ROM directly after reset in modes 1 and 2, and by firmware in the ROM initiated via the control link in mode 3. The memory map and the PMI configuration are described in an ND file and a memory configuration file supplied with the IMS B490 to allow this information to be available to the T9000 toolset products.

Further information on the PMI can be found in [1].

Memory/register	Hardware byte address
64 bit DRAM bank, 4Mbytes	#80000000 to #803FFFFFF ( bank1 )
64 bit DRAM bank, 4Mbytes	#80400000 to #807FFFFFF ( bank2 )
64 bit SRAM bank, 512Kbytes	#80800000 to #8087FFFF ( bank0 )
FIFO data, control, and event control registers	#00000000 to #000007FF ( bank3 )
Firmware ROM, 128Kbytes repeated throughout 16Mbyte space	#7F000000 to #7FFFFFFF( boot bank )

Table 3.2 IMS B490 T9000 memory map

## 5.5 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
<b>VCC, GND</b>		Power supply and return	3,14
<b>ClockIn</b>	in	5MHz clock signal	8
<b>Reset</b>	in	Transputer reset	10
<b>Analyse</b>	in	Transputer error analysis	9
<b>notError</b>	out	Transputer error indicator (inverted)	11
Links			
<b>LinkIn0-3</b>	in	INMOS serial link inputs to transputer	13,5,2,16
<b>LinkOut0-3</b>	out	INMOS serial link outputs from transputer	12,4,1,15
<b>LinkSpeedA,B</b>	in	Transputer link speed selection	6,7

### Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in figure 5.8.

Table 3.3 IMS B490 Pin designations

### 5.5.1 Standard TRAM signals

Signal name	Description
<b>ClockIn</b>	A 5MHz input clock for the transputer. The transputer synthesizes its own high frequency clocks. <b>CikIn</b> should have a stability over time and temperature of 200ppm. <b>CikIn</b> edges should be monotonic within the range 0.8V to 2.0V with a rise/fall time of less than 8ns.
<b>Reset</b>	Resets the transputer, and other circuitry. <b>Reset</b> should be asserted for a minimum of 100ms. After <b>Reset</b> is deasserted a further 100ms should elapse before communication is attempted on any link. After this time, the transputer on this TRAM is ready to accept a boot packet on any of its links. (See figure 5.6)
<b>Analyse</b>	This is used, in conjunction with <b>Reset</b> , to stop the transputer. It allows internal state to be examined so that the cause of an error may be determined. <b>Reset</b> and <b>Analyse</b> are used as shown in figure 5.5. A processor in analyse mode can be interrogated on any of its links.
<b>notError</b>	An open collector output which is pulled low when the transputer asserts its Error pin. <b>notError</b> should be pulled high by a 10K $\Omega$ resistor to <b>VCC</b> . Up to 10 <b>notError</b> signals can be wired together. The combined error signal will be low when any of the contributing signals is low.
<b>LinkOut0-3</b>	Transputer link output signals. These outputs are intended to drive into transmission lines with a characteristic impedance of 100 $\Omega$ . They can be connected directly to the <b>LinkIn</b> pins of other transputers or TRAMs.
<b>LinkIn0-3</b>	Transputer link input signals. These are the link inputs of the transputer. Each input has a 10K $\Omega$ resistor to <b>GND</b> to establish the idle state, and a diode to <b>VCC</b> as protection against ESD. They can be connected directly to the <b>LinkOut</b> pins of other transputers or TRAMs.
<b>LinkSpeedA, LinkSpeedB</b>	These select the speeds of <b>Link0</b> and <b>Link1,2,3</b> respectively. Table 3.4 shows the possible combinations.
<b>Note:</b> <b>Reset</b> resets all hardware on the TRAM. <b>Analyse</b> only affects the on-board IMS T425.	

LinkSpeedA	LinkSpeedB	Link0	Link1,2,3
0	0	10 Mbits/s	10 Mbits/s
0	1	10 Mbits/s	20 Mbits/s
1	0	20 Mbits/s	10 Mbits/s
1	1	20 Mbits/s	20 Mbits/s

Table 3.4 OS link speed selection

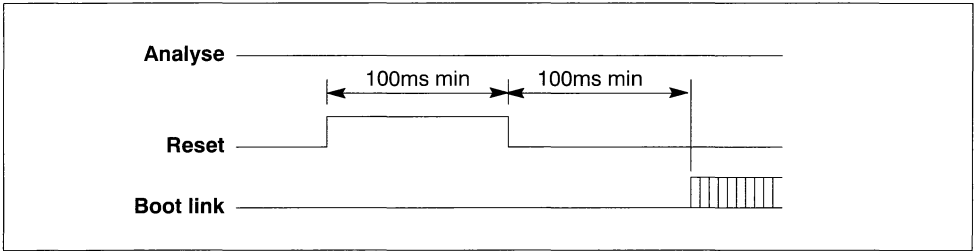


Figure 5.6 Reset timing

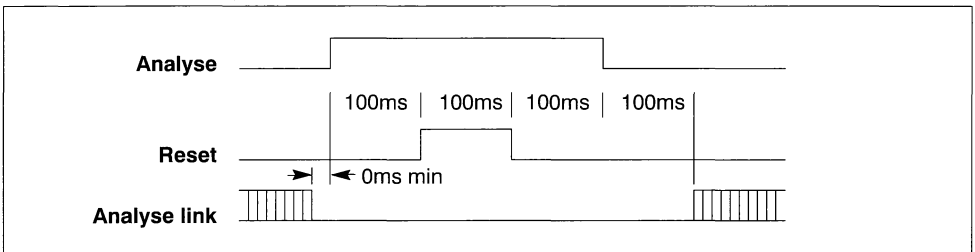


Figure 5.7 Analyse timing

## 5.6 Mechanical details

Figure 5.8 shows an outline drawing of the of the IMS B490.



## 5.7 Installation

Since the IMS B490 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B490 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard below the TRAM slot into which the IMS B490 is to be mounted then the spacer strips should be removed before the TRAM is mounted.

Where the IMS B490 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B490 (see figure 5.8) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot. Plug the IMS B490 into the motherboard.

Should it be necessary to unplug the IMS B490, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B490 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

## 5.8 Specification

TRAM feature		Unit	Notes
Transputer types	T425 and T9000		
Number of transputers	2		
Number of INMOS OS links	4		
Number of INMOS DS-Links	6		1
Amount of SRAM (T9000)	512	Kbyte	
Number of SRAM banks	1		
SRAM bank(s) width	64	bits	
SRAM wait states	0		
Amount of DRAM (T9000)	8	Mbyte	
Number of DRAM banks	2		2
DRAM bank(s) width	64	bits	
DRAM wait states	TBD		
DRAM memory cycle time	TBD	ns	
Subsystem controller	No		
Peripheral circuitry	T9000/T425 interface		
Memory parity	No		
TRAM size	6		
Length	3.66	inch	
Width	6.55	inch	
Pitch between pins	3.30	inch	
Component height above PCB	TBD	mm	
Component height below PCB	2.1	mm	
Weight	TBD	g	3
Storage temperature	0-70	°C	
Operating temperature	10-50	°C	
Airflow	TBD	m/s	
Power supply voltage (Vcc)	4.75-5.25	Volt	
Power consumption (Max)	TBD	W	
Power consumption (Typical)	TBD		

Figure 5.9 IMS B490 specification

### Notes:

- 1 Four Data, control link 0, and control link 1.
- 2 Two banks of 4Mbytes each.
- 3 Weight is approximate and does not include cables.

## 5.9 Ordering Information

To order the IMS B490 TRAM please use the order number in the following table. This order code covers the TRAM, support software for IBM PC and Sun 4, two 0.5m and two 1.5m DS-Link cables, and documentation.

Description	Order Number
IMS B490 TRAM	IMS B490-XX

Figure 5.10 Ordering information

XX refers to processor speed variants. Consult your sales representative for details.

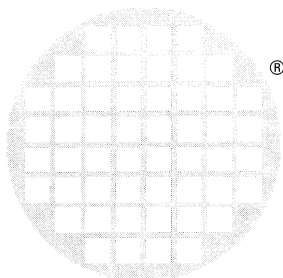
## 5.10 Field Support

INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centers, and authorized distributors.

## 5.11 References

- 1 *T9000 Transputer Hardware Reference Manual*, INMOS Ltd.
- 2 *Module Motherboard Architecture*, INMOS Technical Note 49, INMOS Ltd.
- 3 *Dual-In-Line Transputer Modules (TRAMs)*, INMOS Technical Note 29, INMOS Ltd.
- 4 *The Transputer Development and iQ systems Databook* INMOS Ltd (72-TRN-219-01)
- 5 *T9000 Brochure*, INMOS Ltd 1993





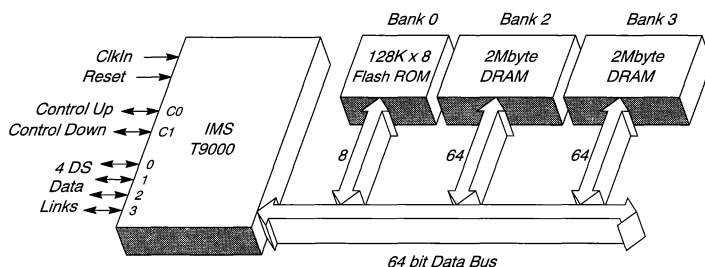
**inmos**®

# IMS B926

## 4Mbyte HTRAM

### Preliminary Information

The information in this datasheet is subject to change



#### FEATURES

- IMS T9000 processor provides 200MIPS and 25MFLOPS (50MHz processor).
- 4Mbytes fast, cacheable DRAM.
- On-board configuration ROM.
- Size 2 HTRAM, compatible with HTRAM motherboards.
- Easily interfaced with other HTRAMs or IMS T9000s to build powerful multiprocessor systems.
- 4 DS data links each provide 100Mbps/s communication with other IMS T9000s or HTRAMs.
- Compilers and development tools available for ANSI C, C++, and occam 2.

#### DESCRIPTION

The IMS B926 is a second generation transputer module (HTRAM), which integrates the high performance IMS T9000 microprocessor with 4Mbytes of fast DRAM. The memory is organised as two 64-bit wide banks, both of which are cacheable. The memory system performs automatic page mode accesses whenever possible to provide maximum memory bandwidth.

An on-board ROM provides for local configuration of the IMS T9000 programmable memory interface and cache, and also contains identification data.

The IMS B926 is part of a family of HTRAMs and HTRAM motherboards and is an ideal building block for multiprocessor systems based on the IMS T9000.

The IMS B926 complies fully with the HTRAM Specification.

## 6.1 Introduction

The IMS B926 is a second generation transputer module (HTRAM), which integrates the high performance IMS T9000 microprocessor with 4Mbytes of fast DRAM. The memory is organised as two 64-bit wide banks, both of which are cacheable. The memory system performs automatic page mode accesses whenever possible to provide maximum memory bandwidth. An on-board ROM provides for local configuration of the IMS T9000 programmable memory interface and cache, and also contains identification data. The IMS B926 is part of a family of HTRAMs and HTRAM motherboards and is an ideal building block for multiprocessor systems based on the IMS T9000. The IMS B926 complies fully with the *HTRAM Specification* (refer to Appendix A).

### 6.1.1 HTRAM Overview

High performance TRAnsputer Modules (HTRAMs) are small assemblies of transputers and other circuits. They have a simple, standard interface, and are designed to be easy to connect together to perform a wide variety of computational and system interface tasks. The circuitry on an HTRAM may consist simply of a transputer and memory, but may also include some interface or special purpose circuitry. A typical interface might be to a disk drive, LAN, or graphics display.

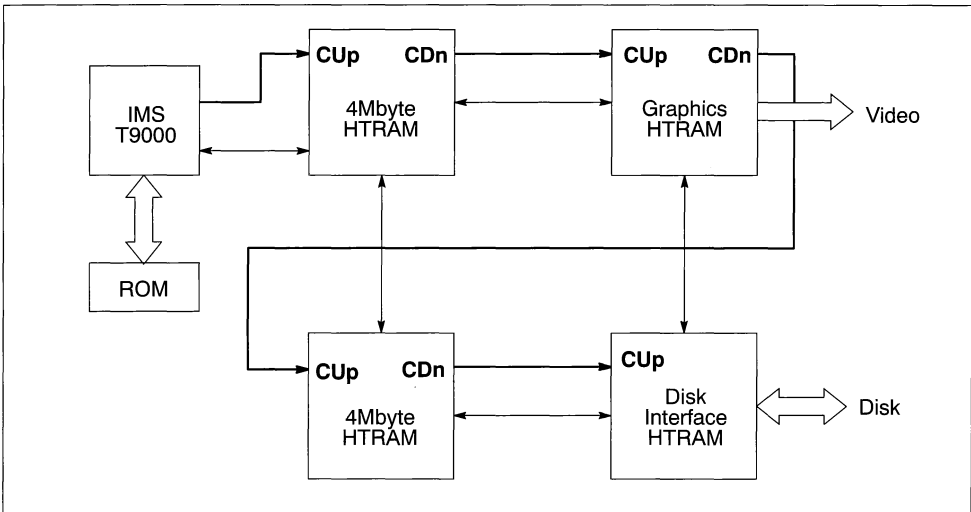


Figure 6.1 Typical stand-alone HTRAM system

Systems are constructed simply by selecting the appropriate HTRAMs, and plugging them into a suitable motherboard. The HTRAMs communicate by means of high speed serial data links.

HTRAMs come in a variety of sizes, based on the unit of the smallest size of HTRAM (size 1), which is about the size of a credit card cut in half lengthwise.

Full information on the HTRAM standard can be found in Appendix A.

### 6.1.2 The IMS T9000 family of devices

The IMS T9000 is the latest member of the transputer family and is designed for embedded systems and high performance computing applications. It integrates a 32-bit integer processor, a 64-bit floating point processor (IEEE 754 compliant), 16Kbytes of cache memory, a communications processor (VCP), and four high bandwidth serial communications links.

Important advances over the previous generation of transputers include:

- Pipelined super-scalar architecture.
- 100Mbit/s serial data links.
- Multiple Virtual Channels on data links between processors.
- 16Kbyte on-chip cache.
- 64-bit wide memory interface.
- Improved error trapping and handling.
- Separate Control and Data link networks.

The family also includes a 32-way packet router: the IMS C104. Multi-processor systems are constructed by connecting transputers and routers together with their communications links.

Full information on the IMS T9000 can be found in [1].

**6.1.3 Communication Links**

The communication links (DS-Links) used by the IMS T9000 family of devices, and by HTRAMs, can operate at up to 100Mbits/s. Each link consists of four signals: **DIn**, **SIn**, **DOut**, **SOut**. A link connection between two devices is made by connecting: **DIn-DOut**, **SIn-Sout**, **DOut-DIn**, **SOut-SIn**. The IMS C100 provides a means of connecting IMS T9000 based systems to systems of previous generation transputers.

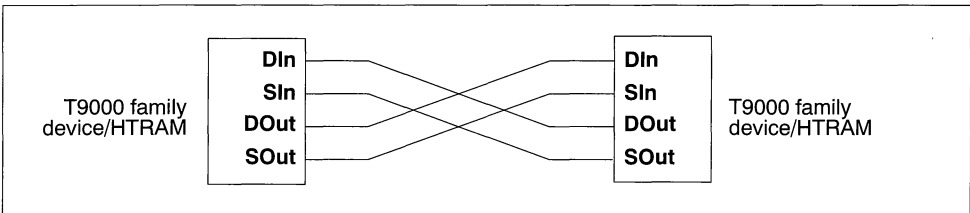


Figure 6.2 DS-Link connection

Link data transmission rates are programmed as part of the system initialization, Link data reception is self-synchronizing. Two devices may communicate even if their data transmission rates are programmed differently, provided that each device is capable of receiving at the transmission rate of the other. Control links do not require configuration.

Full electrical details and a specification of the DS-Link protocol layers can be found in [1].

**6.1.4 Programming Model**

The transputer and HTRAM programming model consists of concurrent processes communicating through channels. Channels connect pairs of processes and allow data to be passed between them. Each process can be built from a number of concurrent processes, so that an entire software system can be described as a process hierarchy. This model is consistent with many modern software design methodologies. The transputer has an efficient, hardware scheduler to manage the execution of concurrent processes.

The component processes of such a software system may run on a single processor or can be distributed across a network of processors. Channels between processes running on different processors are mapped onto the high-speed communication links; an in-built Virtual Channel Processor allows multiple channels to share the same communication link with no software overhead.

**6.1.5 System Behavior**

Application programs are loaded to a network of HTRAMs via their control and data links. Configuration data for the IMS T9000 programmable memory interface is stored in a small ROM on each HTRAM. This

is used to configure the IMS T9000 to get the optimum performance from the memory on the IMS B926. Networks of HTRAMs are supported directly by the T9000 software Toolsets.

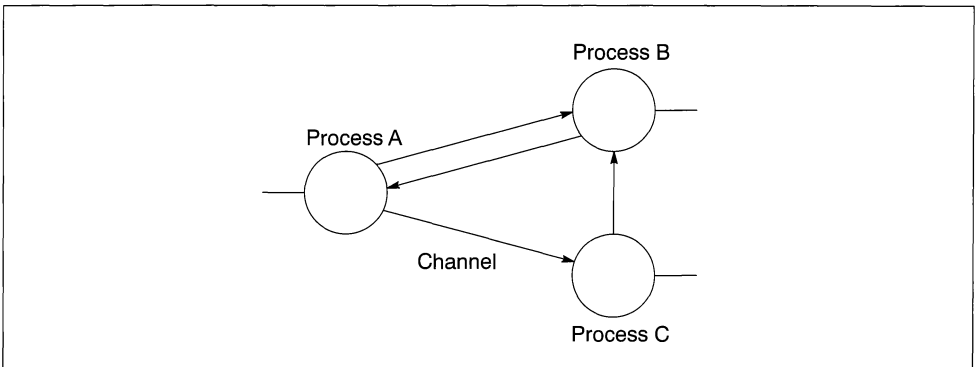


Figure 6.3 Concurrent Processes

### 6.1.6 Software Support

Toolsets are available for developing applications for single and multiprocessor systems in a variety of languages, including ANSI C, C++, and occam 2. The Toolsets contain a comprehensive collection of software development tools, such as:

- Optimizing compilers
- Tools for creating and loading multi-processor programs
- Extensive libraries
- Mixed language programming support
- Powerful debugging and profiling tools for single and multiprocessor systems.

The Toolsets are available for a variety of host systems, including Sun-4 and IBM PC. Please refer to the appropriate data sheets for full details on the T9000 Toolset products, and development platform hardware requirements.

## 6.2 Specification

The IMS B926 is a size 2 HTRAM, which integrates the high performance IMS T9000 microprocessor with 4Mbytes of fast DRAM. The memory is organised as two 64-bit wide banks (connected to banks 2 and 3 of the IMS T9000), both of which are cacheable. The memory system performs automatic page mode accesses whenever possible to provide maximum memory bandwidth. An on-board ROM provides for local configuration of the IMS T9000 programmable memory interface and cache, and also contains identification data.

The IMS B926 complies fully with the *HTRAM Specification* (refer to Appendix A).

### 6.2.1 Interface Signals

The interface signals are as follows. Full electrical details can be found in Appendix A and [1].

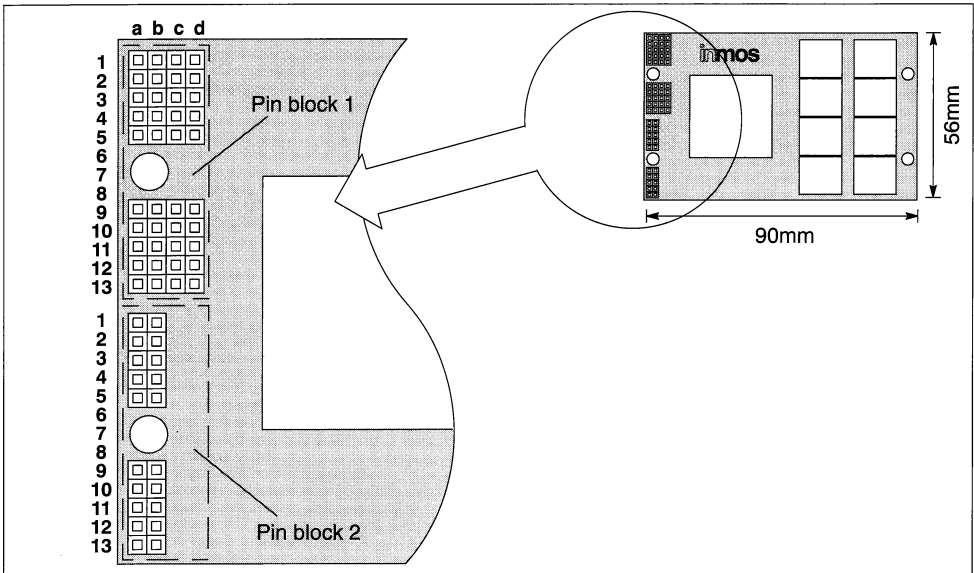


Figure 6.4 Size 2 HTRAM pinout configuration

Pin	Row a	Row b	Row c	Row d
1	CiKIn	N/C	TDI	notTRST
2	L0SIn	GND	V5V0	L2SOut
3	L0DIn	N/C	CUpSIn	L2DOut
4	L0DOut	V3V3	GND	L2DIn
5	L0SOut	N/C	CUpDIn	L2SIn
9	L1SIn	N/C	CUpDOut	L3SOut
10	L1DIn	V5V0	GND	L3DOut
11	L1DOut	N/C	CUpSOut	L3DIn
12	L1SOut	GND	V3V3	L3SIn
13	Reset	TMS	TCK	V5V5
1	N/C	TDO		
2	N/C	GND		
3	N/C	CDnSOut		
4	N/C	V3V3		
5	N/C	CDnDOut		
9	N/C	CDnDIn		
10	N/C	V5V0		
11	N/C	CDnSIn		
12	N/C	GND		
13	N/C	N/C		

Table 4.1 IMS B926 Pinout reference.

Signal name	Description
<b>ClkIn</b>	5MHz clock: the IMS T9000 generates its own high speed clocks.
<b>Reset</b>	IMS T9000 Reset: refer to [1] for a complete description of the IMS T9000 reset behavior.
<b>LxDIn, LxSIn, LxDOut, LxSOut</b>	The IMS B926 has four data links, connected directly to the corresponding IMS T9000 data links. Each data link consists of four signals, for example link 0 consists of <b>L0DIn, L0SIn, L0DOut, L0SOut</b> , and is data link 0 of the IMS T9000. Unused links may be left unconnected. Link connections should be designed as 100Ω transmission lines: no termination is required at either end, as the drivers and receivers are designed to work with unterminated lines.
<b>CUpDIn, CUpSIn, CUpDOut, CUpSOut</b>	The Control Up link is <b>Clink0</b> of the IMS T9000. It is used at system start time to initialize the IMS T9000, and must be connected to a suitable source of control messages: for example the Control Down link of another HTRAM in a daisy chain control architecture.
<b>CDnDIn, CDnSIn, CDnDOut, CDnSOut</b>	The Control Down link is <b>Clink1</b> of the IMS T9000. The Control Down link may be used to drive the Control Up link of another HTRAM or <b>Clink0</b> of an IMS T9000, in a daisy chain control architecture. Control Down may be left unconnected if daisy chain control is not being used, or for the last HTRAM in a control chain.
<b>TMS, TCK, TDI, TDO, notTRST</b>	These form an IEEE1149.1 test access port (TAP). The TAP on the IMS B926 is not implemented: <b>TMS and TCK</b> must be tied high, <b>notTRST</b> must be tied low, <b>TDI</b> should be tied high or connected to the TDO pin of another HTRAM. <b>TDO</b> may be left unconnected or connected to the TDI pin of another HTRAM. Pull-ups/pull-downs of up to 10kΩ may be used.
<b>V5V0</b>	The IMS B926 requires a 5.0V power supply to be connected to the <b>V5V0</b> and <b>GND</b> pins: all of these pins should be connected.
<b>V3V3</b>	The IMS B926 does not require a 3.3V power supply to be connected to the <b>V3V0</b> pins: these pins are not electrically connected on the IMS B926.
<b>GND</b>	Signal reference and power supply return pins.

Table 4.2 Signal descriptions

### 6.2.2 Configuration and ID ROM

All HTRAMs have an ID ROM which can be used to identify the HTRAM in an assembled system. The identification data format is defined in Appendix A. The identification data for the IMS B926 is:

Data Item	Value
IdDataHeader	#44495448
IdDataSize	<i>Subject to change</i>
IdDateVersion	#00000000
VendorString	INMOS Limited
HTRAMtype	IMS B926
Serial Number	<i>Subject to change</i>
NumObjects	#00000000
ObjectTablePtr	#00000000

Table 4.3 ID ROM data

This ROM also contains a configuration program to initialize the PMI bank address, PMI strobe timing, and Cache subsystems of the IMS T9000. The configuration program can be caused to execute by issuing

a Reboot command to the Control Up link of the IMS B926 during system initialization. The configuration program terminates by sending an Error message to the control process. The initialization of other IMS T9000 subsystems is system dependent, and should be made by a controlling process over the control up link.

### 6.2.3 Summary of Features

Feature		Unit
HTRAM type	IMS B926	
Processor type and speed	IMS T9000-xx	
Memory size	4	Mbyte
Cache size	16	Kbyte
Memory organisation	Two 64-bit banks	
Memory cycle time	–	ns
Cache Configuration	from local ROM	
PMI Configuration	from local ROM	
StartFromROM	no	
Test Access Port	inactive	
Application specific pins	none	
HTRAM size	2 (56mm × 90mm)	
Height class	A	
Weight	TBD	g

Table 4.4 Specification

### 6.2.4 Operating Ranges

Functionality is not guaranteed outside the Operating Ranges. Operation beyond the Operating Ranges is not recommended and may affect device reliability.

Parameter	Min.	Typ.	Max.	Unit
Operating temperature	0		50	°C
Airflow	TBD	2		m/s
V5V0	4.75		5.25	V
Input Voltage (any input)	0		V5V0	V
Power consumption (V5V0)			TBD	W
Power consumption (V3V3)		n/a		W

Table 4.5 Operating Ranges

### 6.2.5 Absolute Maximum Ratings

Functionality at or above these limits is not implied. Stresses beyond the Absolute Maximum Ratings may cause permanent damage.

Parameter	Min.	Max.	Unit
Storage temperature	0	70	°C
V5V0 relative to GND	0	7.0	V
Voltage on any pin relative to GND	-0.5	V5V0+0.5	V

Table 4.6 Absolute Maximum Ratings

## 6.3 Ordering Information

Please contact your local sales office or distribution representative for ordering information.

Description	Order Number
4Mbyte HTRAM	IMS B926-XX

Table 4.7 Ordering information

XX refers to processor speed variants. Consult your sales representative for details.

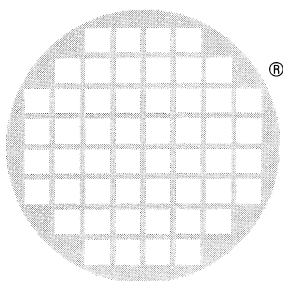
## 6.4 Field Support

INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centers, and authorized distributors.

## 6.5 References

- 1 *T9000 Transputer Hardware Reference Manual*, INMOS Ltd 1993
- 2 *T9000 Brochure*, INMOS Ltd 1993
- 3 *The Transputer Development and iq systems Databook* INMOS Ltd (72-TRN-219-01)





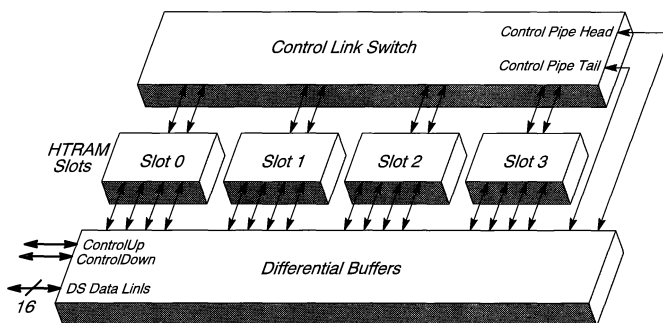
**inmos**<sup>®</sup>

# IMS B100

## A VME Format HTRAM Motherboard

### Preliminary Information

The information in this datasheet is subject to change



#### FEATURES

- Single Slot VME format card
- Accommodates four size 2 HTRAMS
- All 16 Data links from HTRAMs externally accessible
- All external links differentially buffered and brought out to modular DS-Link connectors on the front panel
- Control link pipeline switch

#### DESCRIPTION

VME format HTRAM motherboard with four logical HTRAM slots. Each of the logical slots is size two. The control links from each HTRAM are connected to a Control link switch giving a flexible pipe line architecture. All data links from each HTRAM and the Control pipe line head and tail are differentially buffered and brought out to external INMOS standard DS-Link connectors. Reset facilities are provided by either the VME SYSRESET\* signal or by a push button switch located on the front panel. IMS B100 provides jumpers for the bus grant and interrupt acknowledge signals.

### 7.1 Introduction

The IMS B100 is a member of the family of High performance Transputer module (HTRAM) motherboards. The HTRAM is a second generation transputer module, integrating a high performance IMS T9000 microprocessor with RAM and/or some application specific circuitry. The IMS B100 supports a varying range of HTRAM size configurations. It has 8 physical slots arranged as 4 logical slots, allowing the board to accommodate up to 4 size 2 HTRAMs. All four data DS-Links on every logical slot have been made externally accessible by differential buffers which are then brought out to the front panel via modular DS-Link connectors, allowing a flexible DS-Link communication network. The control link strategy adopted by IMS B100 is that of a re-configurable pipeline architecture, where **ControlUP** and **Control-Down** are differentially buffered and then brought out to the front panel using modular DS-Link connectors. The IMS B100 complies fully with the *HTRAM Specification* (Refer to Appendix A).

#### 7.1.1 HTRAM Overview

HTRAMs are small assemblies of transputers and other circuits. They have a simple, standard interface, and are designed to be easy to connect together to perform a wide variety of computational and system interface tasks. The circuitry on an HTRAM may consist simply of a transputer and memory, but may also include some interface or special purpose circuitry. A typical interface might be to a disk drive, LAN, or graphics display.

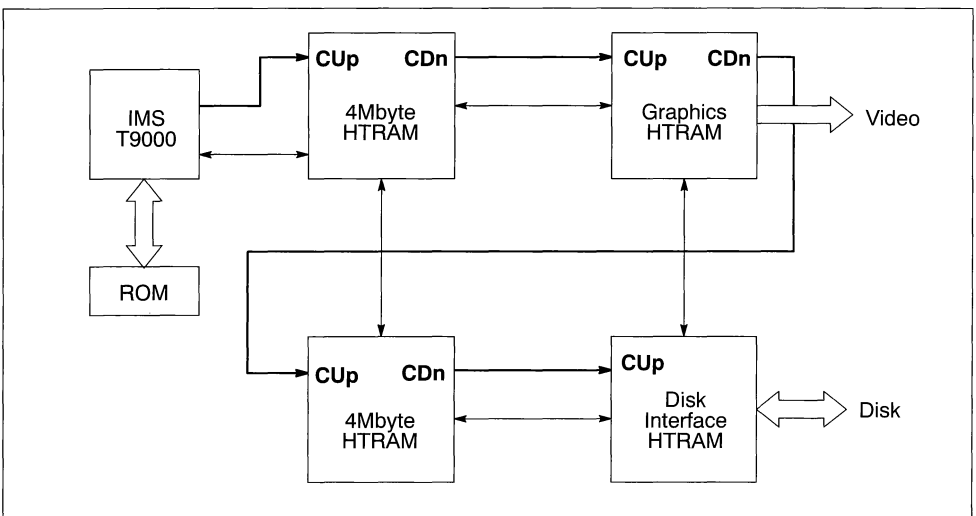


Figure 7.1 Typical stand-alone HTRAM system

Systems are constructed from HTRAMs simply by selecting the appropriate HTRAMs, and plugging them into a suitable motherboard. The HTRAMs communicate by means of high speed serial data links.

HTRAMs come in a variety of sizes, based on the unit of the smallest size of HTRAM (size 1), which is about the size of a credit card cut in half lengthwise.

Full information on the HTRAM standard can be found in Appendix A.

#### 7.1.2 The IMS T9000 family of devices

The IMS T9000 is the latest member of the transputer family and is designed for embedded systems and high performance computing applications. It integrates a 32-bit integer processor, a 64-bit floating point processor (IEEE 754 compliant), 16Kbytes of cache memory, a communications processor (VCP), and four high bandwidth serial communications links.

Important advances over the previous generation of transputers include:

- Pipelined super-scalar architecture.
- 100Mbit/s serial data links.
- Multiple Virtual Channels on data links between processors.
- 16Kbyte on-chip cache.
- 64-bit wide memory interface.
- Improved error trapping and handling.
- Separate Control and Data link networks.

The family also includes a 32-way packet router: the IMS C104. Multi-processor systems can be constructed by connecting transputers and routers together with their communications links.

Full information on the IMS T9000 can be found in [1].

### 7.1.3 Communication Links

The communication links (DS-Links) used by the IMS T9000 family of devices, and by HTRAMs, can operate at up to 100Mbits/s. Each link consists of four signals: **DIn**, **SIn**, **DOut**, **SOut**. A link connection between two devices is made by connecting: **DIn–DOut**, **SIn–SOut**, **DOut–DIn**, **SOut–SIn**. The IMS C100 provides a means of connecting IMS T9000 based systems to systems of previous generation transputers.

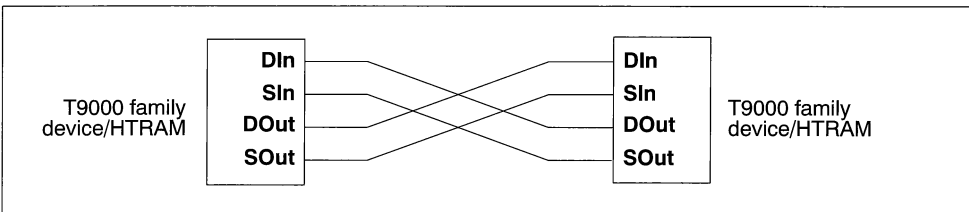


Figure 7.2 DS-Link connection

Link data transmission rates are programmed as part of the system initialization, link data reception is self-synchronizing. Two devices may communicate even if their data transmission rates are programmed differently: provided that each device is capable of receiving at the transmission rate of the other. Control links do not require configuration.

Full electrical details and a specification of the DS-Link protocol layers can be found in [1].

### 7.1.4 Programming Model

The transputer and HTRAM programming model consists of concurrent processes communicating through channels. Channels connect pairs of processes and allow data to be passed between them. Each process can be built from a number of concurrent processes, so that an entire software system can be described as a process hierarchy. This model is consistent with many modern software design methodologies. The transputer has an efficient, hardware scheduler to manage the execution of concurrent processes.

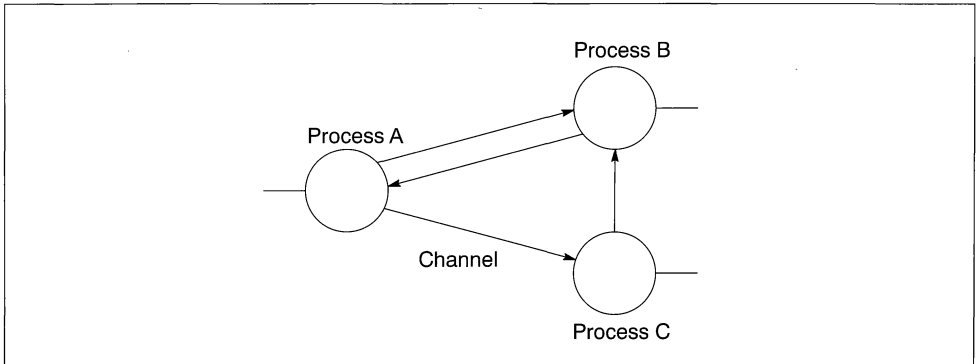


Figure 7.3 Concurrent processes

The component processes of such a software system may run on a single processor or can be distributed across a network of processors. Channels between processes running on different processors are mapped onto the high-speed communication links; an in-built Virtual Channel Processor allows multiple channels to share the same communication link with no software overhead.

Application programs are loaded onto a network of HTRAMs via their control and data links. Configuration data for the IMS T9000 programmable memory interface is stored in a small ROM on each HTRAM. This is used to configure the IMS T9000 to get the optimum performance from the memory on compute HTRAMs. Networks of HTRAMs are supported directly by the T9000 software Toolsets.

### 7.1.5 Software Support

Toolsets are available for developing applications for single and multiprocessor systems in a variety of languages, including ANSI C, C++, and OCCAM 2. The Toolsets contain a comprehensive collection of software development tools, such as:

- Optimizing compilers.
- Tools for creating and loading multi-processor programs.
- Extensive libraries.
- Mixed language programming support.
- Powerful debugging and profiling tools for single and multiprocessor systems.

The Toolsets are available for a variety of host systems, including Sun 4 and IBM PC. Please refer to the appropriate data sheets for full details on the T9000 Toolset products, and development platform hardware requirements.

## 7.2 Hardware Specification

The IMS B100 is a 6U high VME format HTRAM motherboard, requiring only power from a passive VME backplane. There are 8 physical slots provided by the IMS B100, configured as four size 2 logical HTRAM slots. The control link architecture on the IMS B100 has been designed as a re-configurable pipeline, using a control link switch.

The control link switch gives the user the ability to redirect an HTRAM **ControlDown** link to either the next adjacent HTRAM logical slot or to the external **ControlDown** DS-Link connector, thus allowing the control pipeline to bypass vacant logical slots.

On the IMS B100 each logical slot has four DS data links. Each of these links are differentially buffered and then brought out to the front panel using modular DS-Link connectors, thus allowing great flexibility in the data communication networks that can be constructed.

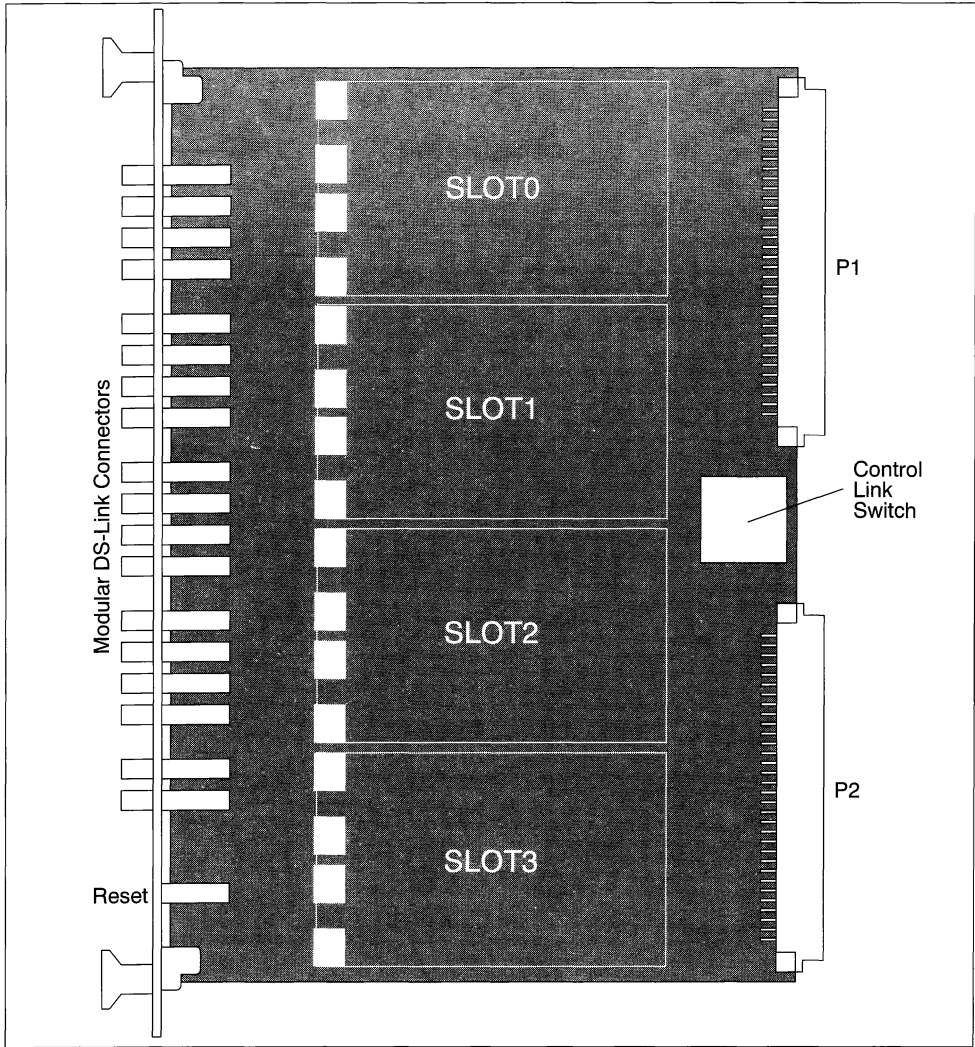


Figure 7.4 IMS B100

### 7.2.1 The Control Links

The IMS B100 incorporates a control link switch, giving the user the ability to re-route the destination of the Control Down link from every logical slot, thus allowing the 'jumping' of inactive logical slots upon the motherboard. This re-configurable architecture allows multiple partially filled motherboards to be placed into a system without breaking the pipeline control link architecture though the system motherboards. Table 5.1 shows the configuration options for the control link switch. The control link switch has the ability to bypass up to three of the HTRAM slots.

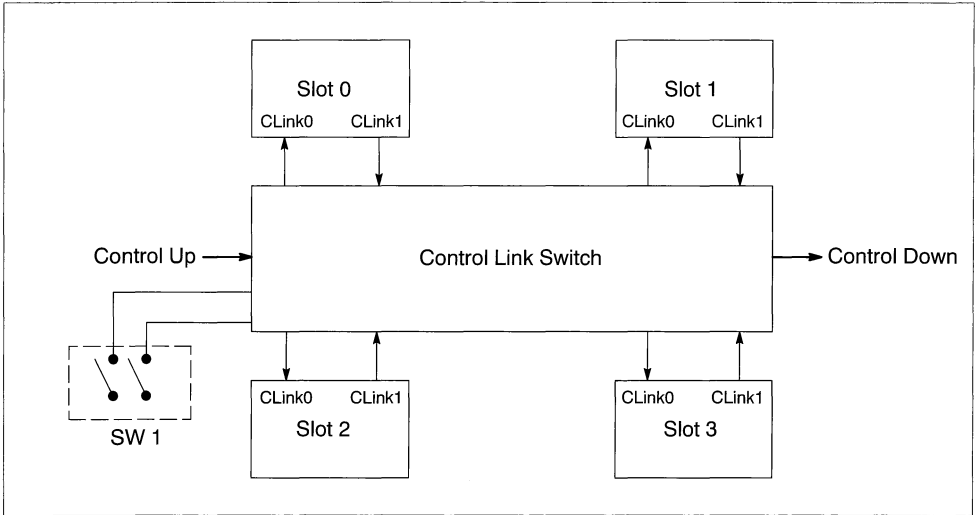


Figure 7.5 Control line pipeline

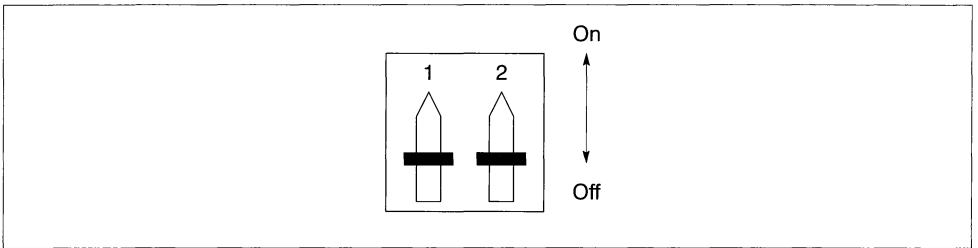


Figure 7.6 Control link switch SW1

SW1:1	SW1:2	Source for External Control Link Down
OFF	OFF	SLOT 0 CLink1
OFF	ON	SLOT 1 CLink1
ON	OFF	SLOT 2 CLink1
ON	ON	SLOT 3 CLink1

Table 5.1 SW1 configuration settings

**7.2.2 External Link connections**

All 18 links, four data links per HTRAM slot and the two control links from the control link switch, are differentially buffered and brought out to connectors. This strategy gives the user great flexibility in designing IMS T9000 communication networks. The differential electrical standard used on the DS-Links of the IMS B100 is compatible with all *iq* systems external DS-Link electrical connections.

**7.2.3 Reset**

A complete reset of the IMS B100 circuitry including HTRAM slots is provided by an on-board power-on reset circuit. The SYSRESET\* can be used as a global system reset, when multiple IMS B100 are plugged into a VME backplane to form large IMS T9000 systems. Otherwise the IMS B100 is supplied with a push button located on the front panel to provide a manual, local, board reset.

**7.2.4 VMEbus**

The IMS B100 does not contain a VMEbus interface. Only VME SYSRESET\* is used by the board. 'Jump-ering' of the **IACK** and **BusGrant** VMEbus signals is provided. No other connections apart from power are made to the VMEbus backplane.

**7.2.5 Power**

The IMS B100 only requires a 5.0 Volt power supply, obtained by use of the VME P1 and P2 backplane connectors. The IMS B100 does not require 3.3 Volts to operate and does not supply 3.3 V to any of the HTRAM slots.

**7.2.6 IMS B100 Front Panel**

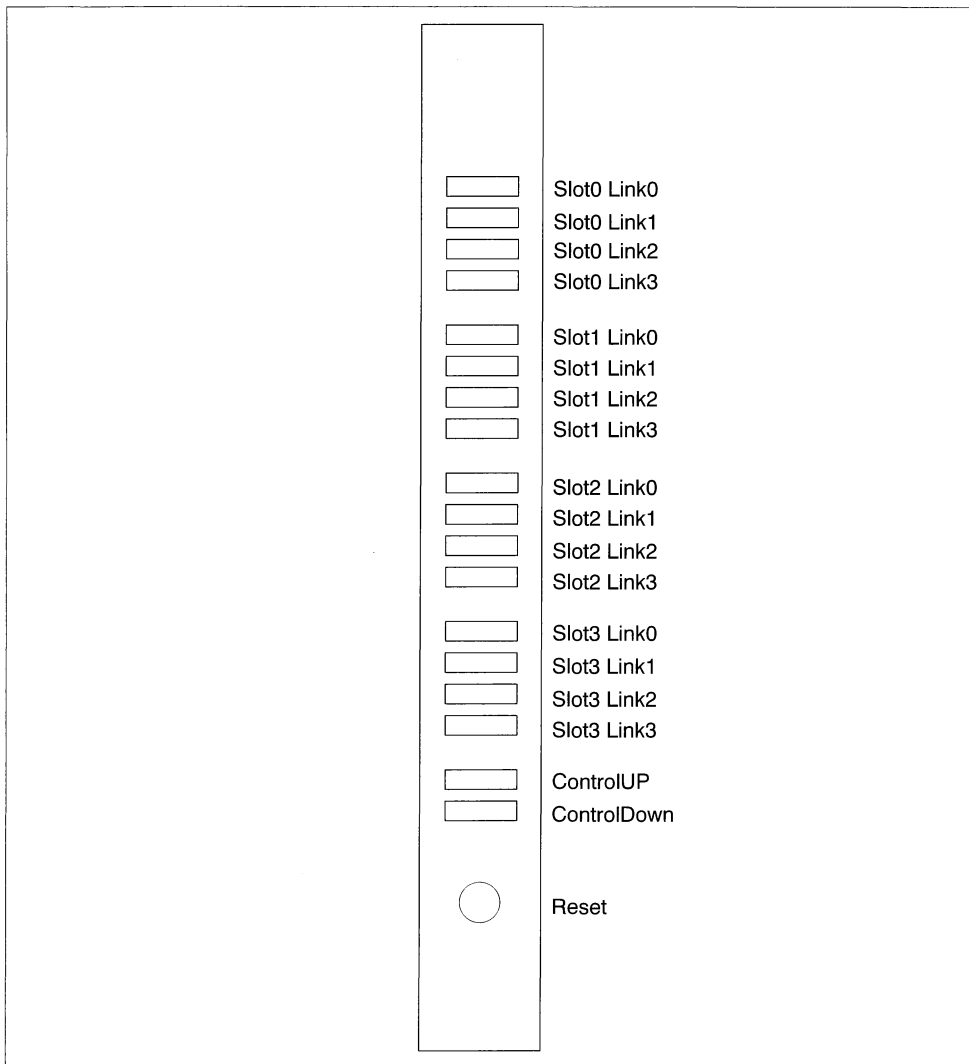


Figure 7.7 Front Panel

### 7.2.7 Mechanical and Thermal details

The IMS B100 is designed in accordance with the VMEbus standard[3]. The overall size of the board is 160mm by 233.35mm with a board thickness of 1.6mm. The supplied front panel width is 4HP, compatible with a board-to-board pitch in a card cage of 0.8". Use of the IMS B100 with HTRAMS of height classes A and B results in an assembly that meets the VMEbus specification for a single width board. Note the front panel is required when operating the IMS B100 in a card cage, both for mechanical rigidity and to maintain correct air flow for cooling.

Adequate air cooling must be provided to ensure that components and HTRAMS are kept within their operating temperature. Failure to do so may affect the reliability of the motherboard and associated HTRAMS. Air flow should run parallel to the board surface and parallel to the front panel.

### 7.2.8 Operating Ranges

Functionality is not guaranteed outside the Operating Ranges. Operation beyond the Operating Ranges is not recommended and may affect device reliability.

Parameter	Min.	Typ.	Max.	Unit
Operating temperature	0		50	°C
Airflow	TBD	2		m/s
+5V DC	4.875		5.25	V
Power consumption (+5V DC)			TBD	W

Table 5.2 Operating Ranges

### 7.2.9 Absolute Maximum Ratings

Functionality at or above these limits is not implied. Stresses beyond the Absolute Maximum Ratings may cause permanent damage.

Parameter	Min.	Max.	Unit
Storage temperature	0	70	°C
Supply Voltage	0	7.0	V

Table 5.3 Absolute Maximum Ratings

## 7.3 Ordering Information

Please contact your local sales office or distribution representative for ordering information.

Description	Order Number
HTRAM motherboard	IMS B100-1

Table 5.4 Ordering information

In addition to the motherboard, an IMS B100 user manual and 5 DS-Link cables (0.5m long) are supplied.

## 7.4 Field Support

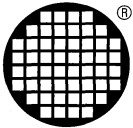
INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centers, and authorized distributors.



## 7.5 References

- 1 *T9000 Transputer Hardware Reference Manual*, INMOS Ltd 1993
- 2 *T9000 Brochure*, INMOS Ltd 1993
- 3 *IEEE Standard for a Versatile Backplane Bus: VMEbus*, IEEE 1987
- 4 *The Transputer Development and iq systems Databook* INMOS Ltd (72-TRN-219-01)





# HTRAM specification

## A.1 Introduction

HTRAMs (High performance TRANsputer Modules) are small assemblies of transputers and other circuitry. They have a simple interface, and are designed to be easy to connect together to perform a wide variety of computational and interface tasks, in hosted or embedded systems.

HTRAMs provide a simple means of building IMS T9000 based single and multi-processor systems. A typical HTRAM system consists of a number of HTRAMs initialized and bootstrapped from a common source, via their control and data links.

The circuitry on an HTRAM may consist simply of a transputer and a high performance memory system, but may also include some kind of interface or special purpose circuitry. A typical interface might be a disk interface, LAN interface, or graphics display driver. Systems are constructed from HTRAMs simply by selecting the appropriate HTRAMs and plugging them into suitable motherboards.

This standard defines the electrical interface, mechanics, environmental requirements, and system behavior of HTRAMs.

## A.2 Interface Signals

### A.2.1 **CIkIn**

All HTRAMs must be provided with a nominal 5MHz clock as specified by Figure A.1 and Table A.1. An HTRAM must present no more than one TTL load to **CIkIn**: if the clock is required by more than one device it must be buffered. **CIkIn** must be received by a device with true TTL input thresholds: devices with CMOS input thresholds must not be used.

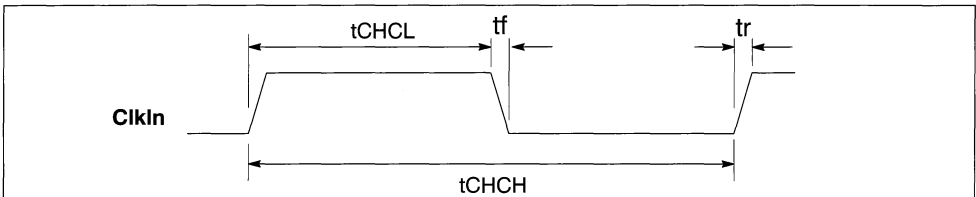


Figure A.1 **CIkIn** Timing

It is recommended that motherboards should drive **CIkIn** as specified by Table A.1, to ensure that **CIkIn** can be received by devices operating on either 5.0V or 3.3V. It is possible to receive these signals with bipolar TTL (e.g. F series), or with 3.3V CMOS devices, or with most BiCMOS TTL families, and with a wide range of PLDs. FCTCT or FBT devices are suitable for driving **CIkIn**.

Symbol	Min.	Typ.	Max.	Units	Notes
VIL	-0.5		0.8	V	
VIH	2.0		3.5	V	
tCHCH		200		ns	1
tCHCL	90	100	110	ns	
trV	TBD			ns	2
tr	2		10	ns	
tf	2		10	ns	

#### Notes:

- 1 **CIkIn** tolerance is  $\pm 100\text{ppm}$ .
- 2 Minimum **Reset** pulse width is with **CIkIn** stable and running.

Table A.1 **CIkIn** and **Reset**

### A.2.2 Reset

A system reset input which is intended to be used as a *level 0 reset* for any T9000 architecture devices on the HTRAM; may also be used to reset any peripheral or control circuitry on the HTRAM. Minimum **Reset** pulse width ( $t_{RV}$ ) is specified by Table A.1. **Reset** is not guaranteed to become active with power up.

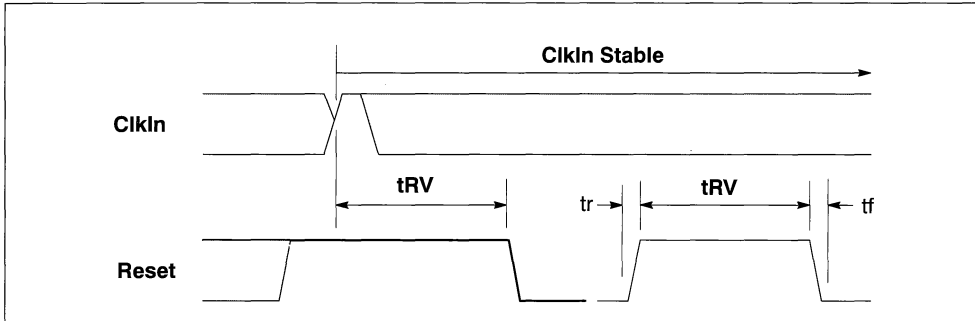


Figure A.2 Reset Timing

An HTRAM must present no more than one TTL load to **Reset**: if it is required by more than one device it must be buffered. **Reset** must be received by a device with true TTL input thresholds: devices with CMOS input thresholds must not be used.

It is recommended that motherboards should drive **Reset** as specified by Table A.1, to ensure that **Reset** can be received by devices operating on either 5.0V or 3.3V. It is possible to receive these signals with bipolar TTL (e.g. F series), or with 3.3V CMOS devices, or with most BiCMOS TTL families, and with a wide range of PLDs. FCTCT or FBT devices are suitable for driving **Reset**.

### A.2.3 DS-Link signals

The HTRAM interface includes two control links and a number of data links. Each link consists of a group of four signals (**DIn**, **SIn**, **DOut**, **SOut**) which carry a single DS (data-strobe) link as defined in the *T9000 Transputer Hardware Reference Manual* [1]. Note that the DIn pins on HTRAMs should be connected to the DIn pins of the T9000 architecture device on the HTRAM; and SIn to SIn, DOut to DOut, SOut to SOut.

The DS-Link inputs must present a high impedance load, and must have TTL input thresholds. The DS-Link input pins of INMOS silicon devices may be used directly, but it is recommended that 10k Ohm pull up resistors should be fitted to suppress noise on unconnected link inputs. The DS-Link outputs must be capable of driving a 100 Ohm transmission line terminated with a high impedance. The DS-Link output pins of INMOS silicon devices may be used directly. Circuit traces for DS-Link signals should be designed as transmission lines with a characteristic impedance of 100 Ohms.

#### Control Up Link

The control up link (**CUpDin**, **CUpSin**, **CUpDout**, **CUpSout**) provides the HTRAM with a source of control information as defined in [1]. All HTRAMs must have a control up link; an HTRAM must not have more than one control up link.

#### Control Down Link

The control down link (**CDnDin**, **CDnSin**, **CDnDout**, **CDnSout**) carries control information as defined in [1]. All HTRAMs must have a control down link; an HTRAM must not have more than one control down link.

#### Use of Control Links

On all HTRAMs, there must be a daisy-chain path from the control up link to the control down link. Typically, this will be provided by connecting the **ClkIn0** and **ClkIn1** pins of all IMS T9000 architecture devices on

the HTRAM in a chain as shown in Figure A.3. The control up link pins should be connected to **CLink0** pins of the first device in the chain. The control down link pins should be connected to the **CLink1** pins of the last device in the chain. If there are no such devices on the HTRAM, the control down link pins must be wired directly to the control up link pins as described by Table A.2.

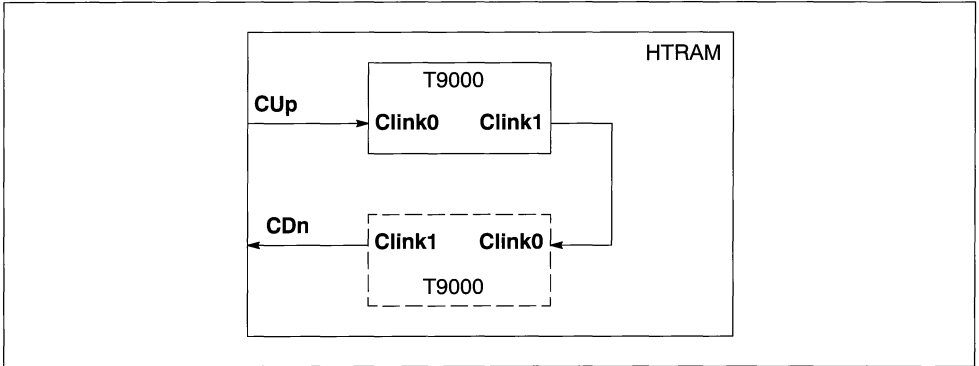


Figure A.3 Control link wiring

CUpDin — CDnDout
CUpSin — CDnSout
CUpDout — CDnDin
CUpSout — CDnSin

Table A.2 Control link wire through

### Control links on motherboards

All HTRAM motherboards must provide a control up link to every logical slot. Note that it is permissible for a motherboard to have fewer logical slots than physical slots. For example, a motherboard may have 32 physical HTRAM slots, of which 16 are provided with a control up link. Such a motherboard can support a maximum of 16 HTRAMs, and is said to have 16 logical slots.

Motherboards may adopt one of two strategies for their control link architecture. Either all logical slots are provided with their own control up link, in which case the control down link pins on the motherboard must be left unconnected; or the HTRAM logical slots should be connected as a daisy chain.

### Data Links

An HTRAM may have up to four DS data links per slot size. Each data link consists of four signals: for example, link0 consists of the signals; **L0Din**, **L0Sin**, **L0Dout**, **L0Sout**. Note that the Din pins on HTRAMs should be connected to the Din pins of the T9000 architecture device on the HTRAM; and Sin to SIn, DOut to DOut, SOut to SOut.

#### A.2.4 Test access port

The pins **TCK**, **TMS**, **TDI**, **TDO**, and **notTRST** are an IEEE1149.1 Test Access Port. This port may be used to implement boundary scan testing of the circuitry on the HTRAM if it contains suitable devices. Use of this feature is not compulsory; but if the test access port is not used **TDI** must be wired through to **TDO**, and **TMS**, **TCK**, and **notTRST** must be left unconnected. HTRAM motherboards are not required to implement this feature; but if it is not used the motherboard must wire **notTRST** to **GND**, **TMS**, **TCK** and **TDI** to **Vcc**, and **TDO** may be left unconnected. Designers intending to use this feature should refer to IEEE1149.1 for full interface details and design information.

Signal	Function
TDI	Test Data In
TDO	Test Data Out
TMS	Test Mode Select
TCK	Test Clock
notTRST	Test Logic Reset

Table A.3 Test Access Port

### A.3 Dimensions

There is a family of HTRAM sizes, based on the dimensions of the size 1 HTRAM. For example, as shown by Figure A.4, a size 2 HTRAM is twice the size of a size 1 and occupies two HTRAM slots on an HTRAM motherboard; a size 4 HTRAM is twice the size again, and occupies four slots on an HTRAM motherboard.

#### A.3.1 Allowed Sizes

All sizes of HTRAM are allowed, including odd number sizes. However, designers should be aware that odd number sized HTRAMs may be incompatible with motherboards which have size 2 logical slots.

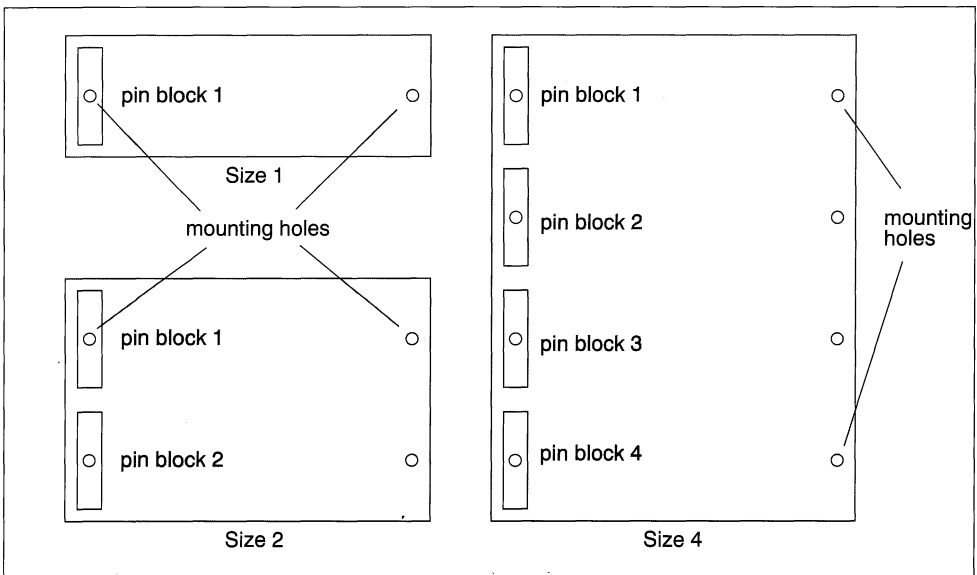


Figure A.4 Example HTRAMs

#### A.3.2 Board Dimensions

The controlling dimensions for a size1 HTRAM are given by Figure A.5. All other dimensions are derived from these, and from the pitch dimension which defines how larger HTRAMs are constructed. The datum point for all linear dimensions is the location of the 1a HTRAM pin.

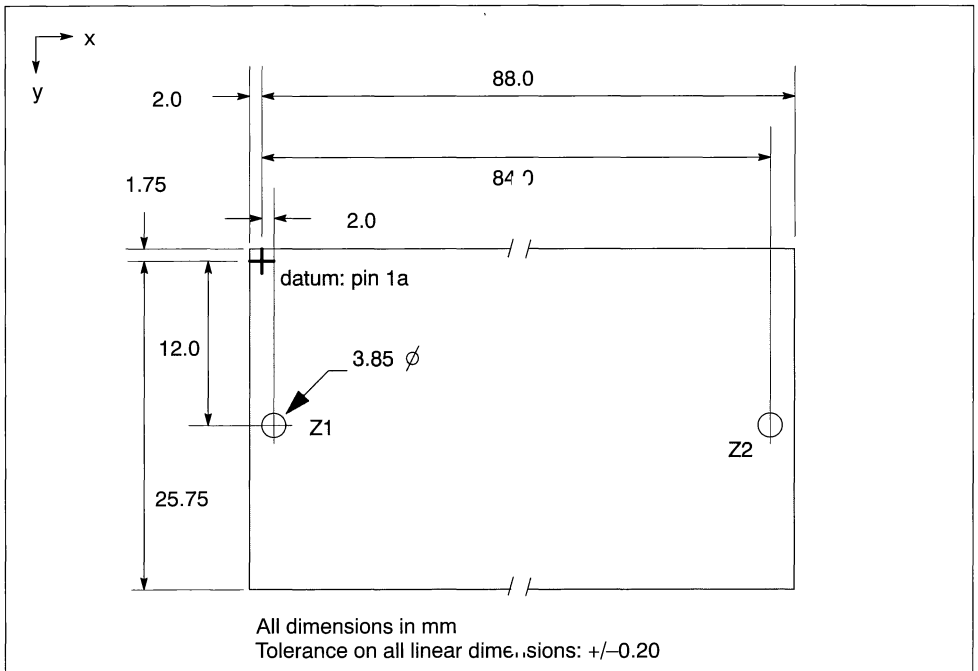


Figure A.5 HTRAM dimensions

The dimensions of larger HTRAMs are derived by placing size 1 HTRAMs adjacent in the short (y) dimension, on a 28.0mm pitch. Figure A.6 is a generalized drawing which gives the dimensions for larger HTRAMs of any size. To use this diagram, N is the size of the HTRAM, k takes all values from 1 to N: figures in square brackets are for a size 2 HTRAM.

If HTRAMS are placed adjacent in the (x) direction, it is recommended that the pitch between boards should be 94.0mm.



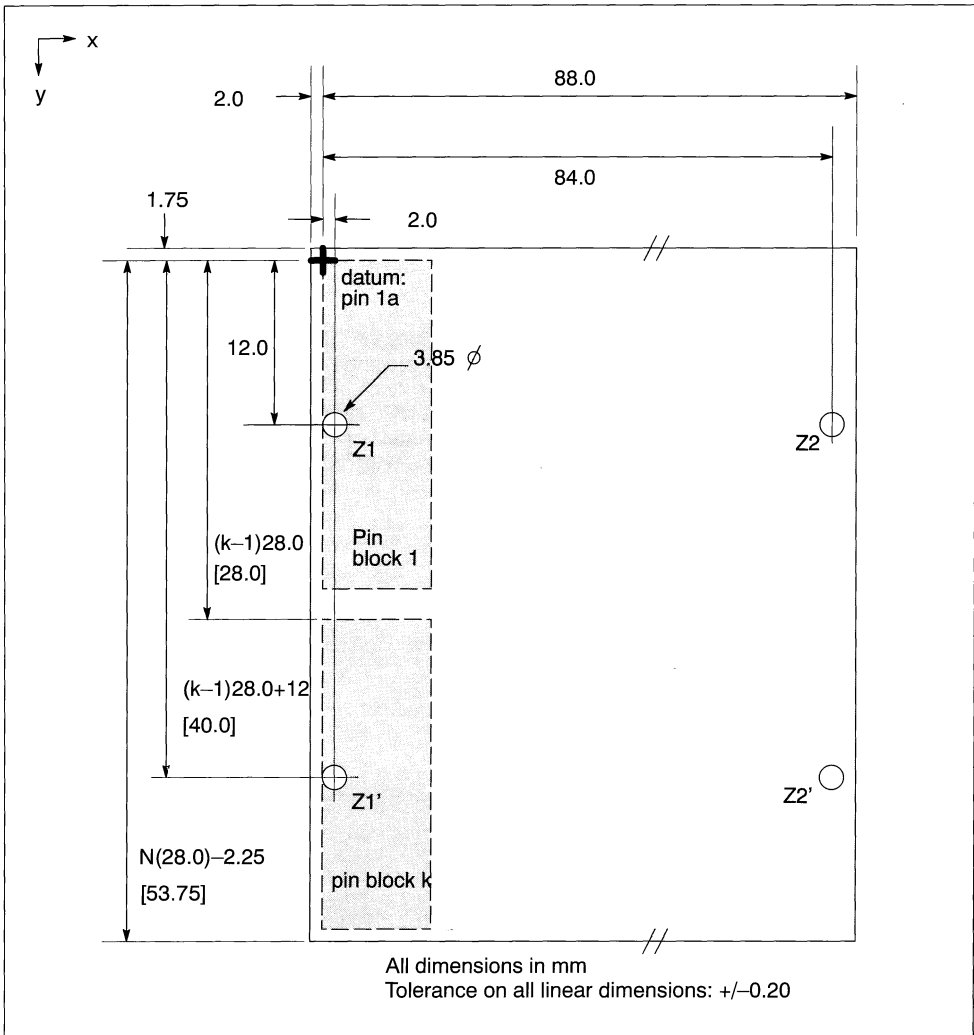


Figure A.6 Generalized HTRAM dimensions

**A.3.3 Fixing Holes**

The signal pins of HTRAMs are not intended to provide mechanical fixing or support. The holes marked **Z1** and **Z2** are intended to fix the HTRAM to the motherboard. It is recommended that some form of mechanical fixing via these holes should always be used. They are suitable for self-locking nylon spacers, which should be adequate for most commercial and industrial applications. Threaded spacers and nuts may be used if excessive vibration prohibits the use of self-locking spacers.

**Location**

Figure A.5 shows the location of fixing holes on a size 1 HTRAM. A size N HTRAM must have 2N fixing holes located as shown by Figure A.6 (except in the case of HTRAMs with IO connectors as described below): the dimension [40.0] is the location of the second set of fixing holes on a size 2 HTRAM.

### Diameter and Clearance

The fixing holes must be unplated, and are nominally 3.85mm in diameter. Components may not be mounted on either surface of the HTRAM PCB within a rectangular area of 6.0mm by 6.0mm centered on each fixing hole.

### HTRAMs with IO connectors

HTRAMs with IO connectors which will be adequately supported by front panel or backplate as described by Section A.3.7, may omit the fixing hole(s) Z2. All other HTRAMs must have fixing holes at both ends.

### A.3.4 Component Placement Area

The top and bottom side component placement areas for a size 1 HTRAM are defined by Figure A.7. The same areas are available relative to pin 1a of each pin block of a larger HTRAM. Note that the component placement area is the same for both the top and bottom surface of the HTRAM PCB, except that the additional area released by omitting pin rows may be used only on the top surface: it may not be used on the bottom surface as components placed there would interfere with the HTRAM sockets on a motherboard. Also note that the available component height on the underside is reduced in the area reserved for application specific interface pins.

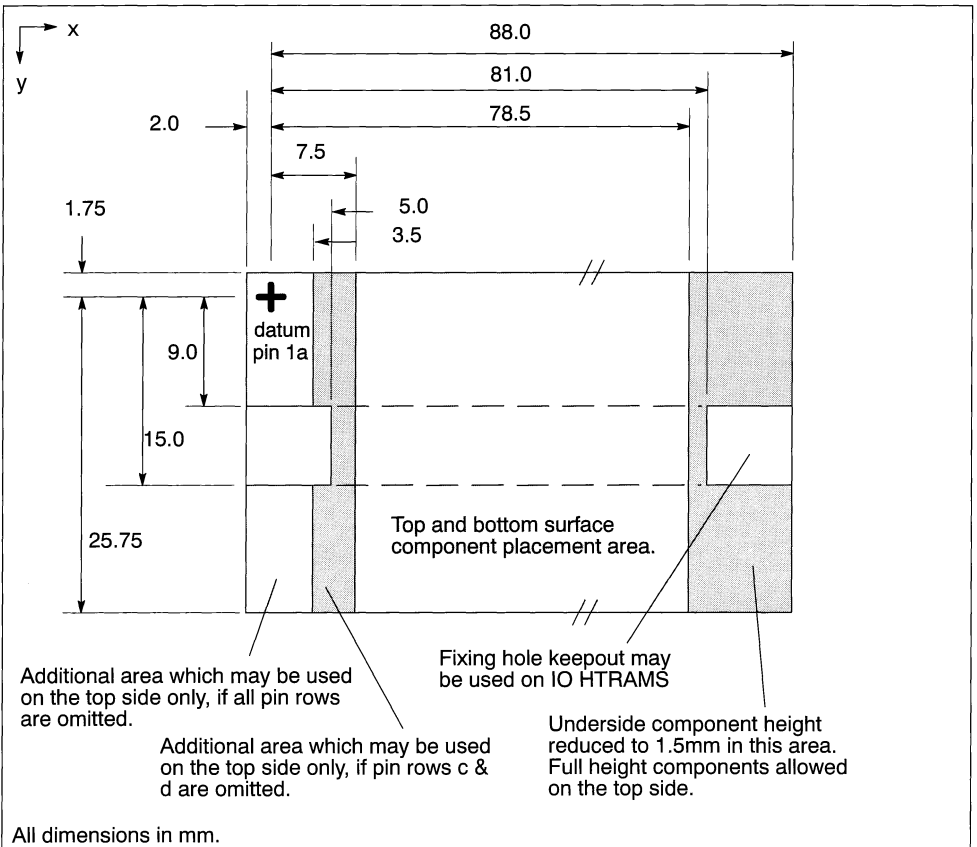


Figure A.7 Component placement area

### A.3.5 Component Height

HTRAMs should be designed to conform to one of the set of height classes defined by Table A.4. HTRAMs designed to conform to these height classes will occupy a single card slot when placed on a motherboard in the given environment. Height class X identifies HTRAMs which mount at extended height, as defined by Section A.3.8. When defining the height class, the height measured must include any mating connectors attached within the component placement area.

This section makes reference to the top side and underside component placement areas defined by Figure A.7. The reference plane for height measurements is the underside surface of the HTRAM PCB.

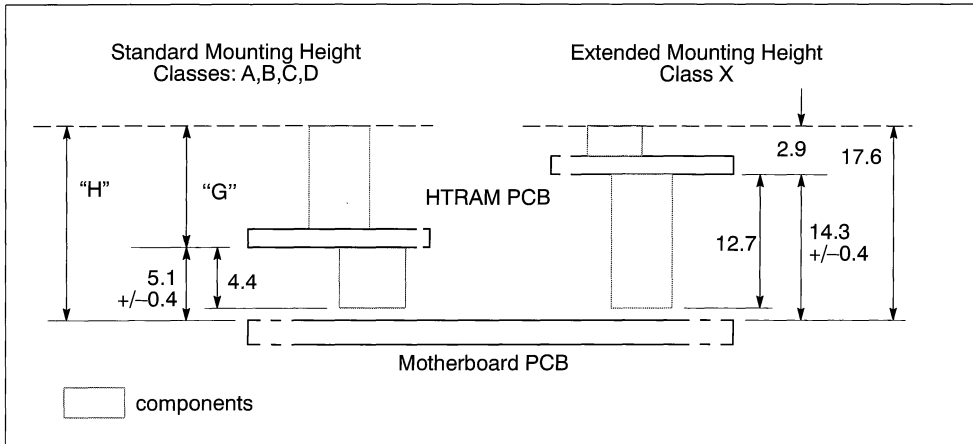


Figure A.8 Allowable Component Heights

#### Recommendations for Standard Mounting Height

- 1 Within the underside component area, components may be mounted to a height of 4.4mm, measured from the underside of the HTRAM PCB. Subject to the additional restrictions for maintaining airflow.
- 2 Within the top side component area, components may be mounted to the height defined by Table A.4, measured from the underside of the HTRAM PCB. Subject to the additional restrictions for maintaining airflow.

#### Recommendations for Extended Mounting Height

- 1 Within the underside component area, components may be mounted to a height of 12.7mm, measured from the underside of the HTRAM PCB. Subject to the additional restrictions for maintaining airflow.
- 2 Within the top side component area, components may be mounted to a height of 2.9mm, measured from the underside of the HTRAM PCB. Subject to the additional restrictions for maintaining airflow.

#### Additional Restrictions for Maintaining Airflow

When viewed in the direction of airflow (y direction, as defined by Figure A.5) components must not obstruct more than 50% of the maximum available cross-section for component mounting.

Height Class	Environment	Dimensions (mm)	
		H	G
A	EISA/ISA	13.34	7.8
B	VME	13.71	8.2
C	PS/2	15.0	9.5
D	P1301	17.6	12.1
X	P1301	17.6	2.9

Table A.4 HTRAM Height Classes

**A.3.6 HTRAM connectors**

HTRAMs are connected to motherboards by blocks of 0.5mm square pins on a 2.0mm by 2.0mm pitch. The socket is fitted to the HTRAM motherboard, the pins are fitted to the HTRAM.

**Pin Blocks**

A size N HTRAM occupies N slot positions on a motherboard. Corresponding to each of these slot positions, the HTRAM has a block of interface pins. These blocks are numbered from 1 to N, as shown by Figure A.4. The pin block number is used to identify pin positions on HTRAMs larger than size 1.

**Pin Positions**

Each pin block consists of a grid of 8 columns (a-h) by 13 rows (1-13) of pins, as shown by Figure A.9. Rows 6, 7, and 8 are occupied by the HTRAM mounting holes, and are not used for electrical pins. Columns (a-d) are used by the standard HTRAM signals. The location of pin 1a is given by Figure A.5.

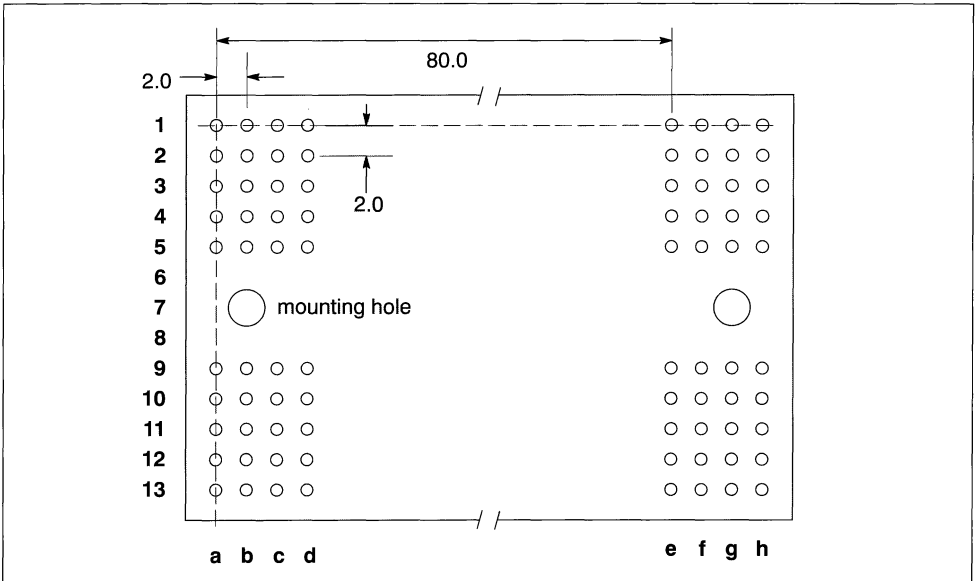


Figure A.9 Detail of pin positions

**Application Specific Pins**

For applications requiring a special purpose interface to the motherboard, HTRAMs may be designed in future with extra signal pins. These are of the same type as the standard HTRAM interface pins and

occupy pin rows e through h as defined by Figure A.9. These pins may be omitted when not used. Motherboards are not required to provide sockets for these pins, but component placement should allow for the presence of sockets in these positions on motherboards, as shown in Figure A.7.

**Connector Dimensions**

Two types of pin header are defined: one for HTRAMs designed to be mounted at standard height, the other for HTRAMs which are designed to be mounted at extended height (IO HTRAMs). Table A.5 details the differences. The same sockets are used on the motherboard for both types.

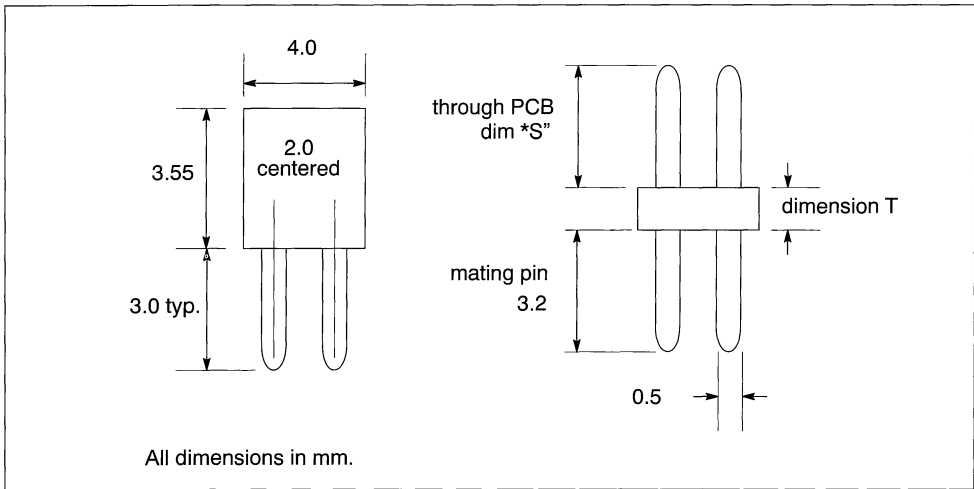


Figure A.10 Connector Dimensions

Mounting Height	PCB Spacing (nom.)	Dimensions (mm)	
		T	S
Standard	5.1	1.5	3.5 (typical)
Extended	14.3	10.7	2.9 (max)

Table A.5 Connector Wafer Thickness

**Recommended Connector**

The Samtec SMM series (socket) is suggested as suitable for motherboards; the TMM series (pin) is suggested as suitable for HTRAMs designed to mount at standard height. Example part numbers are TMM-105-01-S-D for a block of 2 by 5 pins, and SMM-105-01-S-D for a block of 2 by 5 sockets. A longer pin will be required for HTRAMs designed to mount at extended height.

**A.3.7 Connectors for External IO**

Most systems require interfaces to external devices or services, such as disk drives or local area networks. In general, these interfaces require a standard connector which is easily accessible on the outside of the equipment.

If an HTRAM provides an external interface, the connector(s) should be placed at the opposite end of the HTRAM to the HTRAM signal pins. The HTRAM sockets and mounting holes on motherboards should be positioned so that HTRAM pin 1a is 90.0mm from the inside face of the front panel, as shown by Figure A.11. If the connector is correctly positioned on the HTRAM, a good mechanical fit can be achieved between the connector and the front panel attached to the motherboard.

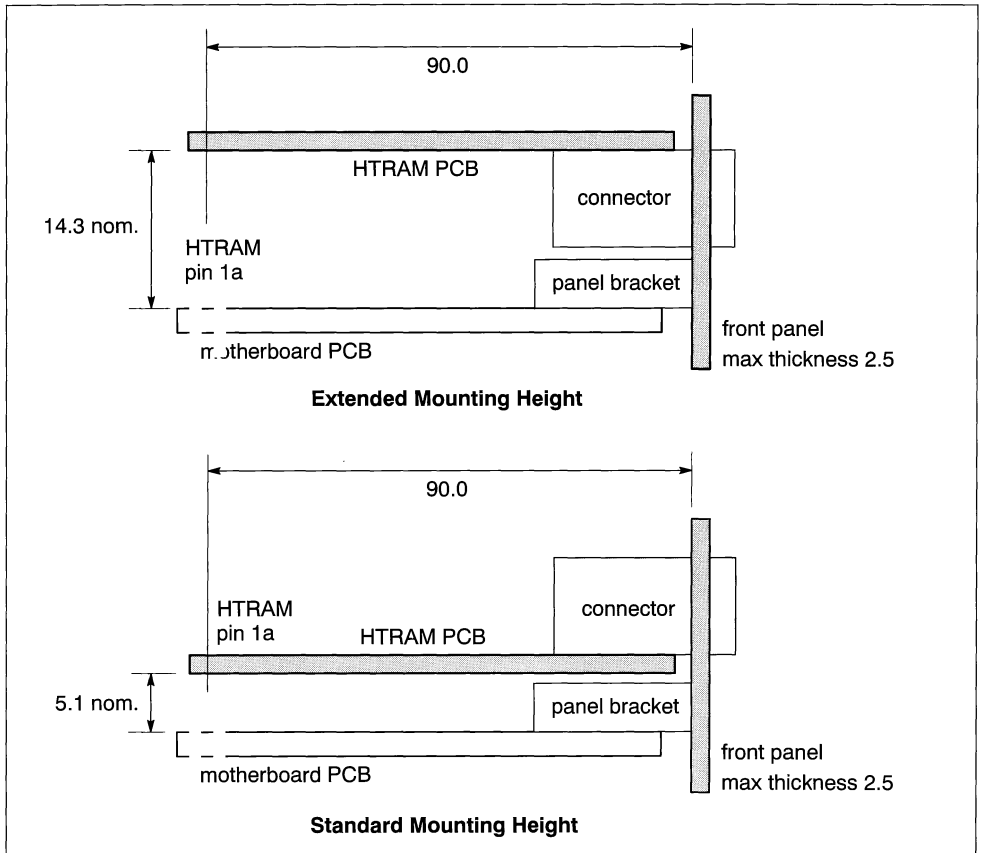


Figure A.11 IO HTRAM Positioning

### A.3.8 Mounting Height

Two mounting heights are defined:

- 1 Standard Height: 5.1mm +/- 0.4mm
- 2 Extended Height: 14.3mm +/- 0.4mm

Use of extended height is restricted to IO HTRAMs; where it is convenient to mount the IO connectors on the underside of the HTRAM so that they fit within the outline of a standard front panel attached to a standard motherboard. All other HTRAMs must be designed with the assumption that they will be mounted at standard height, and must follow the component height recommendations accordingly.

## A.4 Pinout

The pinout of a size1 HTRAM is shown by Figure A.12. This arrangement of pins is replicated in each slot position on a motherboard, but not all pins are connected on HTRAMs larger than size 1. There are some simple rules defining which pins are electrically connected on larger HTRAMs. The electrical signals can be divided into four groups: control signals, daisy chain signals, data links, and power supplies. There is a rule for the location of connecting pins for each signal group.

The rules are designed to ensure that the control up link, **ClkIn**, **Reset**, and the TAP input signals always appears in pin block 1; and that the control down link and **TDO** always appear in pin block N. Thus, on a motherboard which uses a daisy-chain control architecture, the control links and test access port are correctly daisy-chained through HTRAMs larger than size1, without requiring the use of jumpers.

Figure A.13 shows the pinout for an HTRAM of size 2 or larger, derived from the size 1 pinout and the application of these rules. Shaded pins are optional and may be omitted if not required for power.

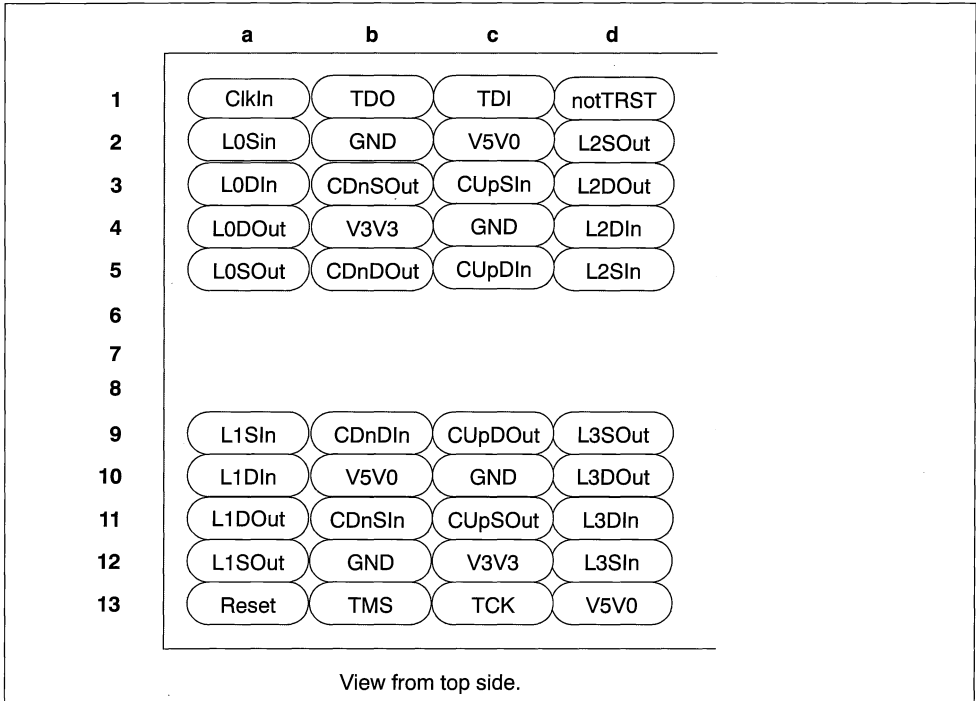


Figure A.12 Pinout of Size 1 HTRAM

**Rule 1: Control Signals**

On all HTRAMs, the signals defined by Table A.6 must be connected only in pin block 1; they must not be connected in any other pin block. This ensures that motherboards can be designed with fewer logical slots than physical slots.

Signal	Pin
CUpDin	c5
CUpSin	c3
CUpDout	c9
CUpSout	c11
ClkIn	a1
Reset	a13
TMS	b13
TCK	c13
TDI	c1
notTRST	d1

Table A.6 Control signals

**Rule 2: Daisy-chain Signals**

On all HTRAMs of size N, the signals defined by Table A.7 must be connected only in pin block N; they must not be connected in any other pin block. Note that for a size 1 HTRAM, pin block N is the same as pin block 1.

Signal	Pin
CDnDin	b9
CDnSin	b11
CDnDout	b5
CDnSout	b3
TDO	b1

Table A.7 Daisy-chain signals

**Rule 3: Data links**

Each pin block provides the connections for four DS data links, so that a size N HTRAM may have up to 4N data links. Most HTRAMs will have only four data links, and these must only be connected in pin block 1. If the HTRAM has a larger number of links, it is recommended that the connections should be made using first the odd numbered pin blocks in order (1, 3, 5 ...), then the even numbered pin blocks in order (2, 4, 6 ...).

Signal	Pin	Signal	Pin
L0Sin	a2	L1Sin	a9
L0DIn	a3	L1DIn	a10
L0DOut	a4	L1DOut	a11
L0SOut	a5	L1SOut	a12
L2Sin	d5	L3Sin	d12
L2DIn	d4	L3DIn	d11
L2DOut	d3	L3DOut	d10
L2SOut	d2	L3SOut	d9

Table A.8 Data link signals



**Rule 4: Power**

Each block of pins provides some power connections. It is the responsibility of the designer to ensure that a sufficient number of power pins are used to supply the maximum power required by the design.

Supply name	Pin
GND	b2, b12, c4, c10
V3V3	b4, c12, d1
V5V0	b10, c2, d13

Table A.9 Power pins

**Rule 5: Optional Pins**

On a size N HTRAM, the pin blocks 2 through N-1 may be omitted unless they are required to provide power: this provides additional space on the HTRAM PCB for mounting components. Pin rows *a*, *c* and *d* may be omitted from block N.

**Rule 6: Use of Pin Blocks 2 through N-1**

No connections, other than power or data links, shall be made through pin blocks 2 through (N-1).

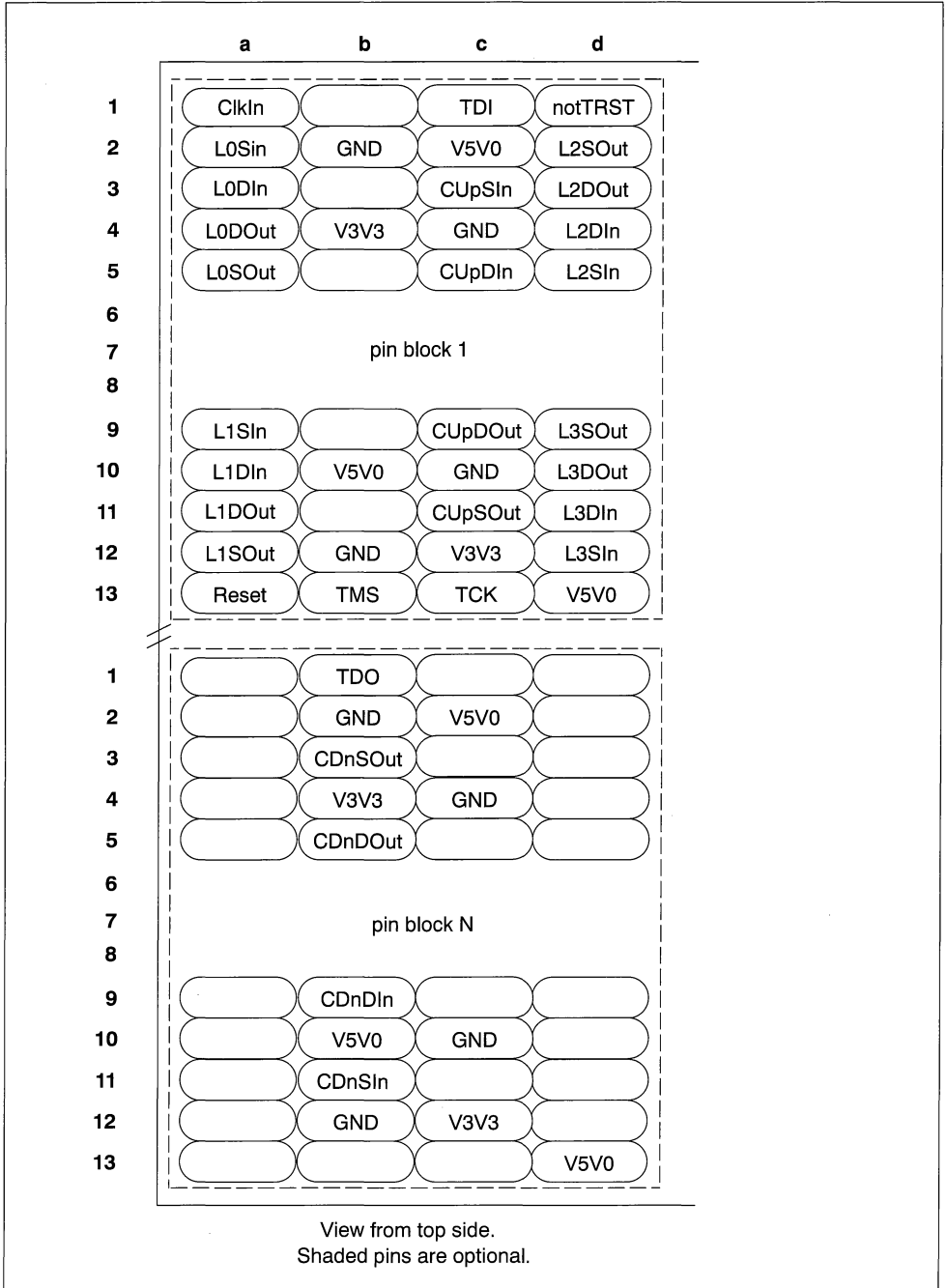


Figure A.13 Pinout of HTRAM larger than size 1

### A.4.1 Provision of sockets on motherboards

All motherboards must provide sockets for pin rows *a–d* in all slot positions. Motherboards must supply power in all slot positions. In the general case, all motherboards will also provide all of the interface signals in all slot positions, and will therefore be capable of supporting all combinations of sizes of HTRAM. However, it is permissible for motherboards to have fewer logical slots than physical slots. For example, a motherboard could be designed with half as many logical slots as physical slots, which was capable of supporting only HTRAMs of size 2, 4, 6 etc. In this case, of each physically adjacent pair of slots on the motherboard, one slot would be wired with the control signals and data links, the other would be wired with the daisy chain signals; so that the wiring of signals on the motherboard would be the same as the wiring of signals on a size 2 HTRAM (Figure A.13).

### A.4.2 Application Specific Pin Allocations

The application specific pins may be allocated non-uniquely; i.e. different application specific interfaces may use the same pins for different purposes. This means that HTRAMs and motherboards which use the application specific pins may not be intermateable. Because of this, designers should think very carefully when designing an HTRAM with an application specific interface. Preliminary pin allocations for application specific interfaces are defined below.

#### Event Pins

If access to the IMS T9000 event pins is required, these must be located as defined by Table A.10.

Pin	Function	Pin	Function
g1	EventIn0	h1	EventOut0
g2	EventIn1	h2	EventOut1
g3	EventIn2	h3	EventOut2
g4	EventIn3	h4	EventOut3

Table A.10 Event pin allocation

### Global Synchronization Pins

Pins *g9–13* and *h9–13* are reserved for future use as global synchronization signals.

## A.5 Power Supplies

Two power supplies are defined, which are nominally 5.0V and 3.3V. Table A.11 details the permitted supply variation. Separate supply pins are provided for each of these, but the return pins (GND) are shared. Each pin is capable of carrying up to 0.5A. The total power which may be consumed from each supply is determined by the number of supply and GND pins fitted. The total drawn from both supplies must not exceed the total capacity of the shared GND pins. Total power dissipation is also limited by the maximum recommended power dissipation. It is the responsibility of the designer to ensure that sufficient power and GND pins are fitted for the maximum power requirements of a design.

Supply name	Min	Typical	Max	Units
V5V0	4.75	5.0	5.25	V
V3V3	3.15	3.3	3.45	V

Table A.11 Supply voltages

### A.5.1 Provision and Use of Supplies

A motherboard is not required to provide a 3.3V supply. However, it is recommended that all motherboards should be designed with provision for the fitting or connection of a 3.3V supply.

Since the first generation of motherboards may not provide a 3.3V supply, it is recommended that the 3.3V supply should not be used unless: the HTRAM is being designed specifically for a 3.3V only environment; or a significant portion of the circuitry requires 3.3V.

Initially, most HTRAMs will operate in a 5.0V only environment. However, it is anticipated that the attraction of reduced power consumption and availability of suitable devices will make mixed and 3.3V only systems possible and desirable. Motherboards for 3.3V only systems are not required to provide a 5.0V supply.

**A.5.2 Maximum Power Dissipation**

An HTRAM should not dissipate more than *TBD* W per slot size.

**A.5.3 Power Classes**

An HTRAM which requires a 5.0V supply is of power class P; an HTRAM which requires a 3.3V supply is of power class Q; an HTRAM which requires both supplies is of power class PQ.

**A.6 Operating Environment**

HTRAMs have the potential to dissipate large amounts of power, which must be removed as waste heat; this requires HTRAMs to be used in an environment with a suitable forced airflow, convection cooling cannot be relied upon. HTRAMs must be designed to operate within the conditions defined by Table A.12.

Parameter	Min	Typical	Max	Units
Ambient temperature	10		52	°C
Airflow	T.B.D.	2.0		m/s
Humidity	10		90	%

Table A.12 Operating environment

**A.6.1 Airflow Requirements**

The preferred direction for cooling air flow is the y direction as defined by Figure A.6. The maximum ambient operating temperature is reduced if the airflow velocity is less than is specified by Table A.12, or is not in the preferred direction.

**A.7 Initialization and Bootstrap**

The HTRAM standard is designed to provide a simple means of building IMS T9000 based single and multiprocessor systems, minimizing the detailed knowledge necessary to get a powerful system up and running. A typical HTRAM system consists of a number of HTRAMs initialized and bootstrapped from a common source, via their control and data links. In order to minimize the HTRAM-specific initialization information that must be included in such a multi-node application, HTRAMs are required to incorporate a uniform method of *local* initialization that can be simply and globally invoked to bring all HTRAMs in a system to a known useful state. This local initialization is achieved through the HTRAM *Configuration ROM*. They are further required to incorporate some identification data in this ROM.

**A.7.1 T9000 Bootstrap**

All IMS T9000 based HTRAMs may be designed to be exclusively controlled and bootstrapped through their external control and data links or they may be designed to operate stand-alone: either as a uniprocessor application, or as the controller of a number of other HTRAM or T9000 based nodes. In either case, the HTRAM configuration ROM must contain enough code and information to correctly initialize the T9000 internal subsystems. This specification distinguishes two cases, according to the state of the StartFrom-ROM pin on the processor.

- **StartFromROM low:** After a hardware reset signal, the processor will await messages on its control up link. Messages from a controller may be used to remotely initialize the T9000 subsystems but, in order to simplify system configuration, the configuration ROM must contain sufficient code to perform this function (and this function only) locally when a Reboot command message is received. This type of ROM is termed a *Local ROM* by the T9000 software development toolset.
- **StartFromROM high:** After a hardware reset signal, the processor will autonomously initiate execution from a ROM that contains both configuration and application code. The type of ROM is termed a *System ROM* by the T9000 software development toolset. As part of the startup sequence or as part of the application, it may take control of and bootstrap other T9000 processors using connections between its data links and their control link network.

It is possible to design an HTRAM that can be switched between these modes (for instance, with a jumper), but the code in the ROM must be able to detect the mode of operation and switch its operation accordingly.

### A.7.2 Configuration ROM contents

An HTRAM configuration ROM is a small, byte-wide memory device which is connected to the Boot Bank of the IMS T9000. It is decoded at processor addresses from #7F000000 to #7FFFFFFF, inclusive.

The configuration ROM must contain both Configuration Code and Identification Data; it may also, optionally, contain a table of named objects. The layout of the ROM is outlined by Figure A.14; its highest memory locations (the *pointer block*) must be as defined by Table A.13.

Address	Size (bytes)	Usage
#7FFFFFFC	4	Reboot Wptr
#7FFFFFF8	4	Reboot lptr
#7FFFFFF4	4	Copy of Reboot lptr (reserved for future use)
#7FFFFFF0	4	Pointer to Identification data (absolute address)

Table A.13 Fixed memory locations in configuration ROM

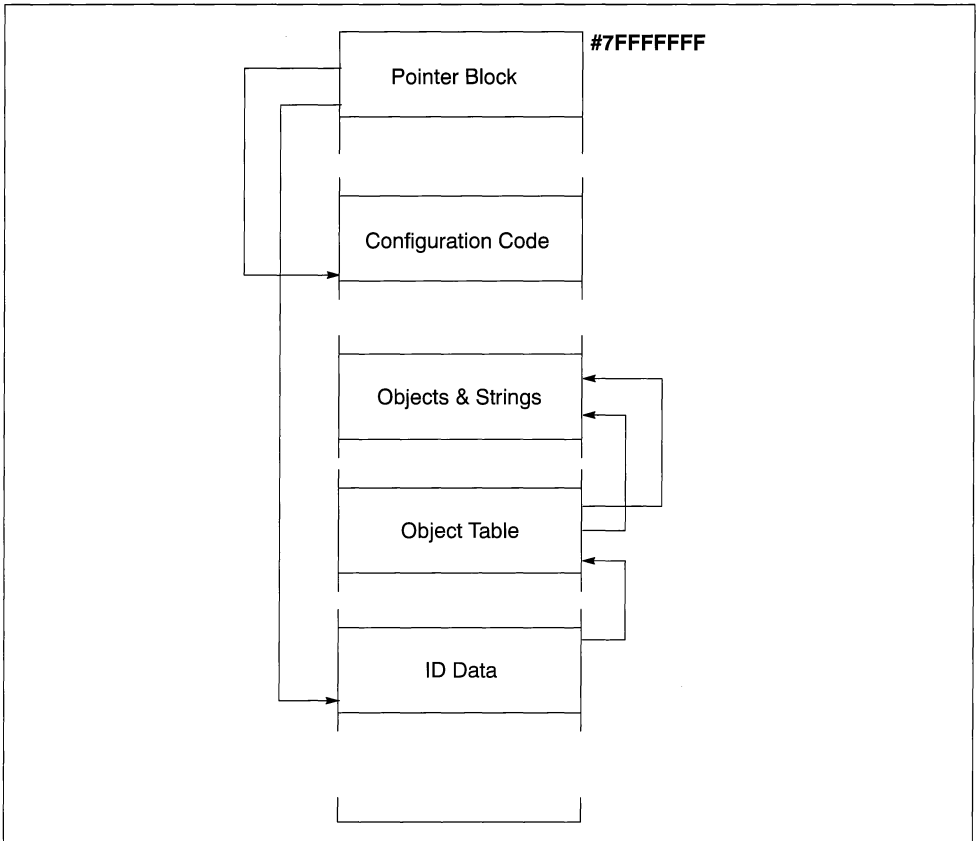


Figure A.14 Configuration ROM

### Configuration Code

The IMS T9000 requires its internal subsystems to be initialized before an application can be loaded or executed. For the Programmable Memory Interface (PMI) and Cache subsystems this initialization is dependent on the type of memory and/or peripheral device attached to the T9000 and without correct initialization the HTRAM will not operate correctly. The Configuration ROM allows this module-specific data to be held on the HTRAM such that initialization of any HTRAM to a known functional state can be achieved by a simple, uniform control link message sequence.

The configuration code must be arranged within the configuration ROM such that it will be executed when the IMS T9000 receives a *Reboot* command on its Control Up link. Even though the IMS T9000 has StartFromROM wired low, receipt of the Reboot command will cause the processor to start execution using *Ip*tr and *Wp*tr values contained in fixed locations at the highest two word addresses in the processor's memory space, as defined by Table A.13. The ROM must contain a valid *Wp*tr and *Ip*tr at these locations, which reference the configuration code. The configuration code must configure the following subsystems within the T9000:

- 1 PMI bank address
- 2 PMI strobe timing
- 3 It may also optionally configure the Cache; it must not write-lock the Cache configuration.

The code should not configure any other subsystems. At the end of this process, the configuration code must do the following:

- 1 Place the signature value "STOP" (#504F5453) in the location referenced by the T9000's *MemStart* configuration register.
- 2 Assert an error to the controlling process (via the control up link) by executing the instruction sequence "causerror 0".

The controlling process then completes the configuration of the T9000's other subsystems via the control up link, and then loads the application to the HTRAM.

### Identification Data

The identification data within the Configuration ROM is required to allow automatic identification of the type and characteristics of any HTRAM within a system. It also contains a serial number that will allow a unique identity to be obtained from any installed HTRAM. The data format also references an *Object Table*, which allows for the presence of any number of arbitrary data items that, at a manufacturers option, may support: additional configuration, self test or peripheral driver executables.

The format of the identification data is defined by Table A.14. The first item listed in the table is at the lowest memory address and it is this (absolute) address that is contained in the identification data pointer defined by Table A.13. All items defined by Table A.14, or referenced by pointer within this data, will be placed at higher addresses within the ROM, as indicated by Figure A.14.

Data item	Size (bytes)	Usage
IdDataHeader	4	Identification data header (#44495448).
IdDataSize	4	Size in bytes of all identification data, from next word to the end of all Objects, inclusive of gaps.
IdDataVersion	4	Identifies the revision of this data format (this version is #00000000).
VendorString	4	Pointer to string containing manufacturers name.
HTRAMtype	4	Pointer to string containing board product code.
SerialNumber	4	Unique serial number of board (or #FFFFFFF).
NumObjects	4	Number of Objects in Object Table (may be 0).
ObjectTablePtr	4	Pointer to Object Table.

Table A.14 Identification Data Format

Note that all strings are stored as sequences of 8 bit ASCII data terminated by a NULL (zero) byte. All numbers are stored in a little-endian fashion: i.e. least significant byte in lowest address. Word alignment of objects is not guaranteed. All pointer values in this data structure are relative to the address of the IdDataHeader.

Note that the ROM must always contain a valid IdDataHeader and IdDataSize, though the IdDataSize field may be 0 if there is no identification data present. Otherwise, the IdDataSize field must be the size of all data, from the IdDataVersion field to the end of the last object or string in the ROM, including all gaps.

### Object Table

The Object Table allows for the inclusion of any number of arbitrary data items that, at a manufacturers option, may support: additional identification, configuration, self test or peripheral driver executables. Any number of objects is permitted, but each must have a unique name string. Objects may contain code or data. The format of each entry in the Object Table is defined by Table A.15.

Item	Size (bytes)	Usage
ObjectNamePtr	4	Pointer to string containing name of Object.
ObjectSize	4	Size in bytes of Object data.
ObjectPtr	4	Pointer to Object data itself.

Table A.15 Object Table Format

Object Table entries must be contiguous in memory, the ObjectNamePtr occupying the lower memory address. The ObjectTablePtr points to the ObjectNamePtr in the first entry in the Object Table.

### A.7.3 Defined Objects

The following objects are defined at present. They need not exist in every HTRAM Configuration ROM but if they are present, the Object data must have a content compatible with that described below:

- "IMS\_htram\_NDL" – contains an ASCII representation of the NDL source file used to generate the configuration code.
- "IMS\_htram\_memconfig" – contains a copy of the memory configuration data used to generate the configuration code.

Object Names beginning with "IMS" are reserved for objects defined by INMOS Ltd. All objects defined by INMOS will have names of this form. It is recommended that other HTRAM vendors adopt a similar naming convention.

## A.8 Specification of HTRAMs

It may be helpful to adopt a standard specification table of the form given by Table A.16. A shorthand notation is defined for describing the size and power supply requirements of an HTRAM. For example, a size 2 HTRAM of height class B, which requires a 5.0V only supply, is described as: size 2B, power class P; or 2BP. If it was also an IO HTRAM it would be described as 2BP1.

Feature		Units	Notes
HTRAM type	IMS Bxxx		
Processor type and speed	IMS T9000 –30		
Memory size	4	Mbytes	
Memory cycle time	80	ns	
Memory organization	Two 64-bit banks		
IO HTRAM	YES/NO		
Test Access Port	YES/NO		
Application specific circuitry	SCSI interface		
Application specific interface pins	NONE		
Size	2		
Height class <sup>a</sup>	B		1
Power class	P		

- 1 When defining the height class, the height measured must include any mating connectors attached within the component placement area.

**Note:** All parameters in this table are examples and do not describe any particular product.

Table A.16 Example Specification Table

## A.9 References

- 1 *T9000 Transputer Hardware Reference Manual*, INMOS Ltd, 1993.