inmos®

TRANSPUTER DATABOOK

Third edition 1992

SGS-THOMSON
MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics Group

INMOS Databook Series

Transputer Databook
Military and Space Transputer Databook
Transputer Development and *i*q Systems Databook
Transputer Applications Notebook: Architecture and Software
Transputer Applications Notebook: Systems and Performance
T9000 Transputer Products Overview Manual
Graphics Databook

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 is a member of the SGS–THOMSON Microelectronics Group.

INMOS document number: 72 TRN 203 02

ORDER CODE: DBTRANST/3

PRINTED IN ITALY

Contents overview

Contents

Notation and nomenclature

The nomenclature and notation in general use throughout this databook is described below.

Significance

The bits in a byte are numbered 0 to 7, with bit 0 least significant. The bytes in words are numbered from 0, with byte 0 least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

Similarly, components of arrays are numbered starting from 0 and stored in memory with component 0 at the lowest address.

Transputer memory is byte addressed, with words aligned on four-byte boundaries for 32 bit devices and on two-byte boundaries for 16 bit devices.

Hexadecimal values are prefixed with #, as in *#1DF*.

Where a byte is transmitted serially, it is always transmitted least significant bit (0) first. In general, wherever a value is transmitted as a number of component values, the least significant component is transmitted first. Where an array is transmitted serially, component 0 is transmitted first. Consequently, block transfers to and from memory are performed starting with the lowest (most negative) address and ending with the highest (most positive) one.

In diagrams, the least significant component of a value is to the right hand side of the diagram. Component 0 of an array is at the bottom of a diagram, as are the most negative memory locations.

Signal naming conventions

Signal names identifying individual pins of a transputer chip have been chosen to avoid being cryptic, giving as much information as possible. The majority of transputer signals are active high. Those which are active low have names commencing with **not**; names such as **RnotW** imply that the first component of the name refers to its active high state and the second to its active low state. Capitals are used to introduce new components of a name, as in **ProcClockOut**.

All transputer signals described in the text of this databook are printed in **bold**. Registers and flags internal to a device are also printed in **bold**; instruction operation codes are printed in *italics*. *Italics* are also used for emphasis. occam program notation is printed in a `fixed space teletype` style.

References

The databook is divided into *chapters* each containing a number of *sections* and *subsections*. Figures and tables have reference numbers tied to relevant sections of a particular chapter and unless otherwise stated refer to sections within the current chapter.

Examples

Software and hardware examples given in this databook are outline design studies and are included to illustrate various ways in which transputers can be used. The examples are not intended to provide accurate application designs.

Transputer product numbers

Product numbers take the following form:

IMS abbbc-xyyz

IMS = INMOS company identifier

a = Product group

T = Transputer
C = Communications peripheral
S = general software
D = Development software
F = Application software
B = Motherboards and TRAMs

bbb = Unique product identifier

e.g. 805 = 32 bit transputer, FPU, 4K memory, 4 links.

c = Revision code

This is not present on all products.
Product traceability is guaranteed by a separate lot number found elsewhere on the package.

x = Package type

G = PGA
P = Plastic DIL
S = Ceramic DIL
J = PLCC
F = Ceramic QFP
P = Plastic QFP
N = Ceramic LCC
E = Plastic SOJ

yy = Speed variant

z = Specification

S = Commercial 0–70°C
E = Extended –55/+125°C
I = Industrial –40/+85°C
M = Mil Std 883C –55/+125°C

(Note: **x**, **yy** and **z** apply to product groups T and C only.)

Company overview

1 INMOS

INMOS is a recognised leader in the design and development of high performance integrated circuits and is a pioneer in the field of parallel processing. Components are designed and manufactured to satisfy the most demanding of current processing applications and also provide an upgrade path for future applications.

INMOS has a consistent record of innovation over a wide product range and supplies components to system manufacturing companies in the United States, Europe, Japan and the Far East. As developers of the transputer, a unique microprocessor concept with a revolutionary architecture, and the occam parallel processing language, INMOS has established the standard for the future exploitation of parallel processing. INMOS also manufactures graphics devices including color look-up tables (CLUTs), color video controllers (CVCs), and XGA chip sets.

INMOS is a member of the SGS-THOMSON Microelectronics Group, a major supplier of a wide range of semiconductor devices. Operating as a division of SGS-THOMSON, INMOS services its customers through the SGS-THOMSON sales network.

2 SGS-THOMSON Microelectronics

SGS-THOMSON Microelectronics is an international microelectronics company born in 1987 as a result of the merger between Thomson Semiconducteurs and SGS Microelettronica. In April 1989 SGS-THOMSON strengthened its position on the international scene by acquiring INMOS.

90% of SGS-THOMSON's capital is controlled by IRI/Finmeccanica (Italy) and by Thomson CSF (France) on an equal basis; the remaining 10% is held by Thorn EMI (UK), former owner of INMOS. The Chief Executive Officer of SGS-THOMSON Microelectronics is Pasquale Pistorio.

The Group has over 17,000 employees, 8 research and development units, 24 design centers, 17 production plants, 50 sales offices in 20 different countries, and over 500 distributors and representatives worldwide.

In 1991 SGS-THOMSON, with sales close to 1.5 billion dollars, confirmed its position as second largest European semiconductor supplier. Some 20% of total turnover is annually reinvested in research and development, and SGS-THOMSON takes an active part in all the European programs for technological research such as JESSI and Eureka.

The SGS-THOMSON product portfolio includes over 20,000 products ranging from simple diodes and transistors to digital and linear integrated circuits, memories and microprocessors. The Company holds the following positions:

Number 2 worldwide for EPROM memories, with a complete range of products from low density up to 16 Mbit devices. Its market share for these leading products is approximately 10% worldwide and 5% in Japan.

Number 1 worldwide in power integrated circuits e.g. audio amplifiers, with 1 billion ICs sold, and motor control devices.

Number 2 worldwide for analogue telecommunications ICs.

Number 2 worldwide for linear ICs for the automotive market.

Number 1 in Europe in both the telecommunications and industrial applications markets.

Number 2 in the European consumer and automotive sectors.

Semiconductors control the technological advancement and the performance of all electronic systems; in order to guarantee technological independence to its shareholders and to its European and worldwide customers, SGS-THOMSON intends to continue the development of a very wide product range capable of covering all application sectors of advanced electronics. It will also continue to reinforce cooperation links and strategic alliances both with customers and with other semiconductor producers.

3 Introduction to transputers

The transputer family is a well established range of 16 and 32 bit microprocessors. Launched in 1985, there are now six major processor variants and three peripherals offered in a variety of speed options and advanced surface mount packages, all detailed in this book. The T9000 transputer is described in a separate book, the '*IMS T9000 Hardware Reference Manual*'.

Over 500,000 units have been shipped to customers worldwide and the transputer ranks as one of the leading microprocessor architectures.

A transputer is a single VLSI device with processor, memory and serial communications links for direct connection to other transputers. Concurrent systems can be constructed from a collection of transputers operating in parallel and communicating through the links, with occam as the associated design formalism. Transputers can also be programmed in languages such as C, C++, FORTRAN and ADA.

The INMOS serial communications link is a high speed system interconnect which provides full duplex communication between members of the transputer family. It can also be used as a general purpose interconnect even where transputers are not used. The IMS C011 and IMS C012 link adaptors are communications devices enabling the INMOS serial communications link to be connected to parallel data ports and microprocessor buses. The IMS C004 is a programmable link switch. It provides a full crossbar switch between 32 link inputs and 32 link outputs.

The transputer is ideally suited to embedded systems, combining low system cost with a powerful CPU and the unique capability of its serial communications links. From designs employing a single transputer to systems containing thousands, the transputer has given its users the ability to manufacture leadership products ahead of the competition.

4 Quality and reliability

Transputers are manufactured to rigorous quality standards backed by an advanced program of training, statistical quality control, and quality improvement teams throughout the company. Full details are contained in the SGS-THOMSON '*SURE*' catalogue, available from SGS-THOMSON sales offices and authorized distributors worldwide.

5 Military products

A range of parts processed to Mil Std 883 Class B are available. Further details are contained in the '*Military and Space Transputer Databook*'.

6 Development systems

The success of any microprocessor is dependent on its development tools and the transputer has an unrivalled set of mature products successfully used by thousands of developers worldwide.

INMOS software products include integrated compilers and toolsets for ANSI C, C++, FORTRAN and occam, supported on IBM PC, NEC PC, SUN and VAX based systems. Each compiler is supported by a toolset containing all the standard tools such as debuggers and simulators. Mixed language development and portability of code across hosts are fully supported. Third parties offer support for ADA, Modula 2, and many other languages.

INMOS hardware products have been designed to be modular and are based on Motherboards such as PC, VME, and Eurocard, which contain slots for transputer modules (TRAMs). TRAMs may be compute modules (transputer + memory), or application related (e.g. graphics, image capture, SCSI, GPIB, flash ROM and many more). The TRAM standard is fully documented as an open standard, now adopted by many companies worldwide.

The products are guaranteed to a full quality and reliability standard. Further details are contained in the '*Transputer Development and* **iq** *systems Databook*'.

Transputer architecture

1 Introduction

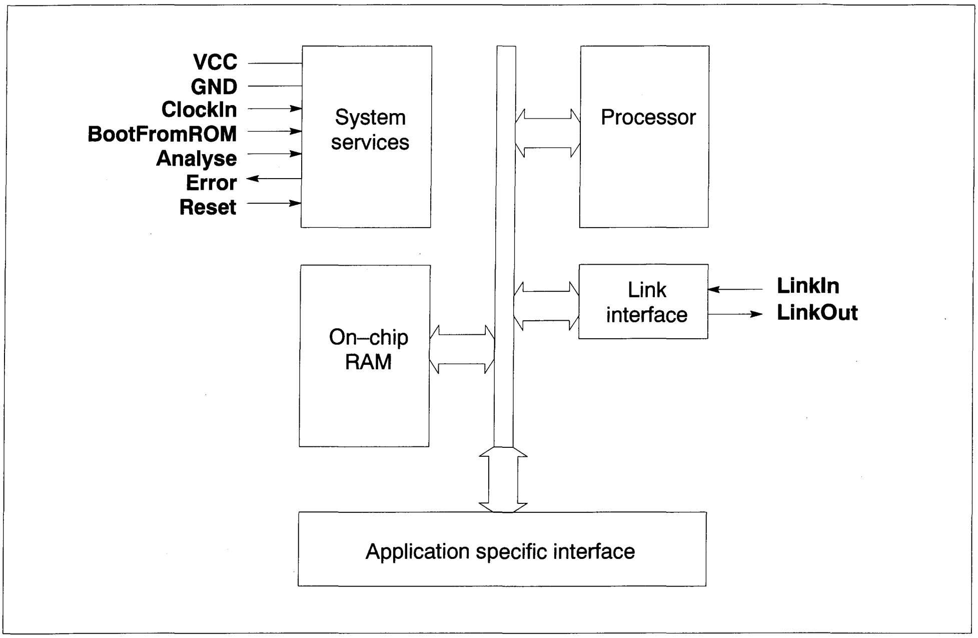


Figure 1.1 Transputer architecture

1.1 Overview

A transputer is a microcomputer with its own local memory and with links for connecting one transputer to another transputer.

The transputer architecture defines a family of programmable VLSI components. The definition of the architecture falls naturally into the *logical* aspects which define how a system of interconnected transputers is designed and programmed, and the *physical* aspects which define how transputers, as VLSI components, are interconnected and controlled. A typical member of the transputer product family is a single chip containing processor, memory, and communication links which provide point to point connection between transputers. In addition, each transputer product contains special circuitry and interfaces adapting it to a particular use. For example, a peripheral control transputer, such as a graphics or disk controller, has interfaces tailored to the requirements of a specific device.

A transputer can be used in a single processor system or in networks to build high performance concurrent systems. A network of transputers and peripheral controllers is easily constructed using point-to-point communication.

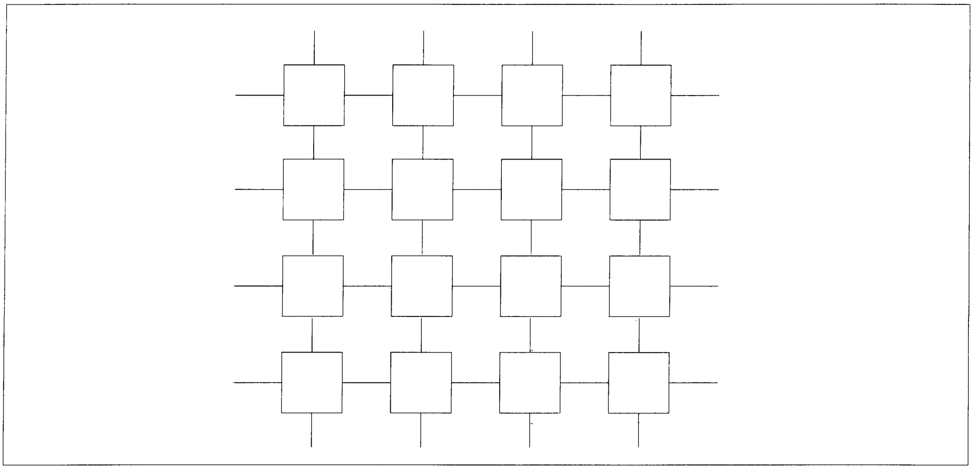


Figure 1.2 Transputer network

Transputers and occam

Transputers can be programmed in most high level languages, and are designed to ensure that compiled programs will be efficient. Where it is required to exploit concurrency, but still to use standard languages, occam can be used as a harness to link modules written in the selected languages.

To gain most benefit from the transputer architecture, the whole system can be programmed in occam (page 11). This provides all the advantages of a high level language, the maximum program efficiency and the ability to use the special features of the transputer.

occam provides a framework for designing concurrent systems using transputers in just the same way that boolean algebra provides a framework for designing electronic systems from logic gates. The system designer's task is eased because of the architectural relationship between occam and the transputer. A program running in a transputer is formally equivalent to an occam process, so that a network of transputers can be described directly as an occam program.

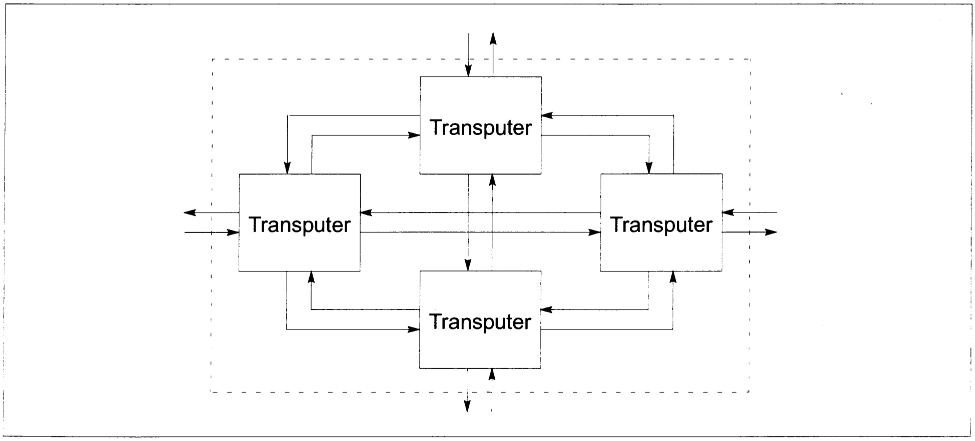


Figure 1.3 A node of four transputers

1.2 System design rationale

The transputer architecture simplifies system design by the use of processes as standard software and hardware building blocks.

An entire system can be designed and programmed in occam, from system configuration down to low level I/O and real time interrupts.

1.2.1 Programming

The software building block is the *process*. A system is designed in terms of an interconnected set of processes. Each process can be regarded as an independent unit of design. It communicates with other processes along point-to-point channels. Its internal design is hidden, and it is completely specified by the messages it sends and receives. Communication between processes is synchronized, removing the need for any separate synchronization mechanism.

Internally, each process can be designed as a set of communicating processes. The system design is therefore hierarchically structured. At any level of design, the designer is concerned only with a small and manageable set of processes.

occam is based on these concepts, and provides the definition of the transputer architecture from the logical point of view (page 11).

1.2.2 Hardware

Processes can be implemented in hardware. A transputer, executing an occam program, is a hardware process. The process can be independently designed and compiled. Its internal structure is hidden and it communicates and synchronizes with other transputers via its links, which implement occam channels.

Other hardware implementations of the process are possible. For example, a transputer with a different instruction set may be used to provide a different cost/performance trade-off. Alternatively, an implementation of the process may be designed in terms of hard-wired logic for enhanced performance.

The ability to specify a hard-wired function as an occam process provides the architectural framework for transputers with specialized capabilities (e.g., graphics). The required function (e.g., a graphics drawing and display engine) is defined as an occam process, and implemented in hardware with a standard occam channel interface. It can be simulated by an occam implementation, which in turn can be used to test the application on a development system.

1.2.3 Programmable components

A transputer can be programmed to perform a specialized function, and be regarded as a 'black box' thereafter. Some processes can be hard-wired for enhanced performance.

A system, perhaps constructed on a single chip, can be built from a combination of software processes, pre-programmed transputers and hardware processes. Such a system can, itself, be regarded as a component in a larger system.

The architecture has been designed to permit a network of programmable components to have any desired topology, limited only by the number of links on each transputer. The architecture minimizes the constraints on the size of such a system, and the hierarchical structuring provided by occam simplifies the task of system design and programming.

The result is to provide new orders of magnitude of performance for any given application, which can now exploit the concurrency provided by a large number of programmable components.

1.3 Systems architecture rationale

1.3.1 Point to point communication links

The transputer architecture simplifies system design by using point to point communication links. Every member of the transputer family has one or more standard links, each of which can be connected to a link of some other component. This allows transputer networks of arbitrary size and topology to be constructed.

Point to point communication links have many advantages over multi-processor buses:

There is no contention for the communication mechanism, regardless of the number of transputers in the system.

There is no capacitive load penalty as transputers are added to a system.

The communications bandwidth does not saturate as the size of the system increases. Rather, the larger the number of transputers in the system, the higher the total communications bandwidth of the system. However large the system, all the connections between transputers can be short and local.

1.3.2 Local memory

Each transputer in a system uses its own local memory. Overall memory bandwidth is proportional to the number of transputers in the system, in contrast to a large global memory, where the additional processors must share the memory bandwidth.

Because memory interfaces are not shared, and are separate from the communications interfaces, they can be individually optimized on different transputer products to provide high bandwidth with the minimum of external components.

1.4 Communication

To provide synchronized communication, each message must be acknowledged. Consequently, a link requires at least one signal wire in each direction.

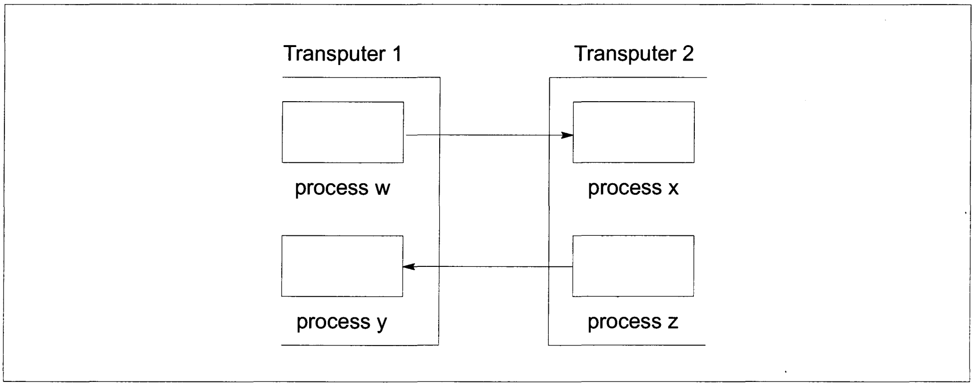


Figure 1.4 Links communicating between processes

A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer by two one-directional signal lines, along which data is transmitted serially.

The two signal wires of the link can be used to provide two occam channels, one in each direction. This requires a simple protocol. Each signal line carries data and control information.

The link protocol provides the synchronized communication of occam. The use of a protocol providing for the transmission of an arbitrary sequence of bytes allows transputers of different word length to be connected.

Each message is transmitted as a sequence of single byte communications, requiring only the presence of a single byte buffer in the receiving transputer to ensure that no information is lost. Each byte is transmitted as a start bit followed by a one bit followed by the eight data bits followed by a stop bit. After transmitting a data byte, the sender waits until an acknowledge is received; this consists of a start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged byte, and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

Data bytes and acknowledges are multiplexed down each signal line. An acknowledge can be transmitted as soon as reception of a data byte starts (if there is room to buffer another one). Consequently transmission may be continuous, with no delays between data bytes.

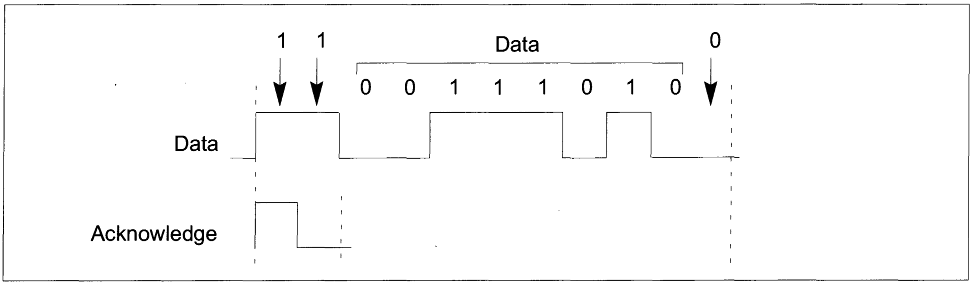


Figure 1.5 OS link protocol

The links are designed to make the engineering of transputer systems straightforward. Board layout of two wire connections is easy to design and area efficient. All transputers will support a standard communications frequency of 10 Mbits/sec, regardless of processor performance. Thus transputers of different performance can be directly connected and future transputer systems will directly communicate with those of today.

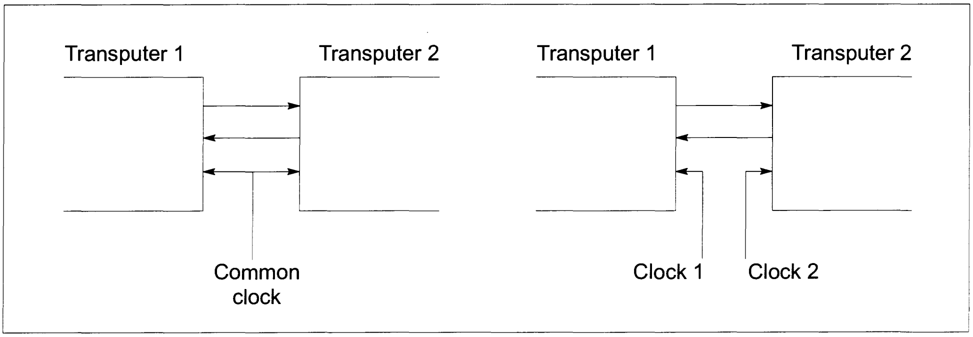


Figure 1.6 Clocking transputers

Link communication is not sensitive to clock phase. Thus, communication can be achieved between independently clocked systems as long as the communications frequency is the same.

The transputer family includes a number of link adaptor devices which provide a means of interfacing transputer links to non-transputer devices.

2 occam model

The programming model for transputers is defined by occam. The purpose of this section is to describe how to access and control the resources of transputers using occam. A more detailed description is available in the occam programming manual and the transputer development system manual (provided with the development system).

The transputer development system will enable transputers to be programmed in other industry standard languages. Where it is required to exploit concurrency, but still to use standard languages, occam can be used as a harness to link modules written in the selected languages.

2.1 Overview

In occam processes are connected to form concurrent systems. Each process can be regarded as a black box with internal state, which can communicate with other processes using point to point communication channels. Processes can be used to represent the behavior of many things, for example, a logic gate, a microprocessor, a machine tool or an office.

The processes themselves are finite. Each process starts, performs a number of actions and then terminates. An action may be a set of sequential processes performed one after another, as in a conventional programming language, or a set of parallel processes to be performed at the same time as one another. Since a process is itself composed of processes, some of which may be executed in parallel, a process may contain any amount of internal concurrency, and this may change with time as processes start and terminate.

Ultimately, all processes are constructed from three primitive processes – assignment, input and output. An assignment computes the value of an expression and sets a variable to the value. Input and output are used for communicating between processes. A pair of concurrent processes communicate using a one way channel connecting the two processes. One process outputs a message to the channel and the other process inputs the message from the channel.

The key concept is that communication is synchronized and unbuffered. If a channel is used for input in one process, and output in another, communication takes place when both processes are ready. The value to be output is copied from the outputting process to the inputting process, and the inputting and outputting processes then proceed. Thus communication between processes is like the handshake method of communication used in hardware systems.

Since a process may have internal concurrency, it may have many input channels and output channels performing communication at the same time.

Every transputer implements the occam concepts of concurrency and communication. As a result, occam can be used to program an individual transputer or to program a network of transputers. When occam is used to program an individual transputer, the transputer shares its time between the concurrent processes and channel communication is implemented by moving data within the memory. When occam is used to program a network of transputers, each transputer executes the process allocated to it. Communication between occam processes on different transputers is implemented directly by transputer links. Thus the same occam program can be implemented on a variety of transputer configurations, with one configuration optimized for cost, another for performance, or another for an appropriate balance of cost and performance.

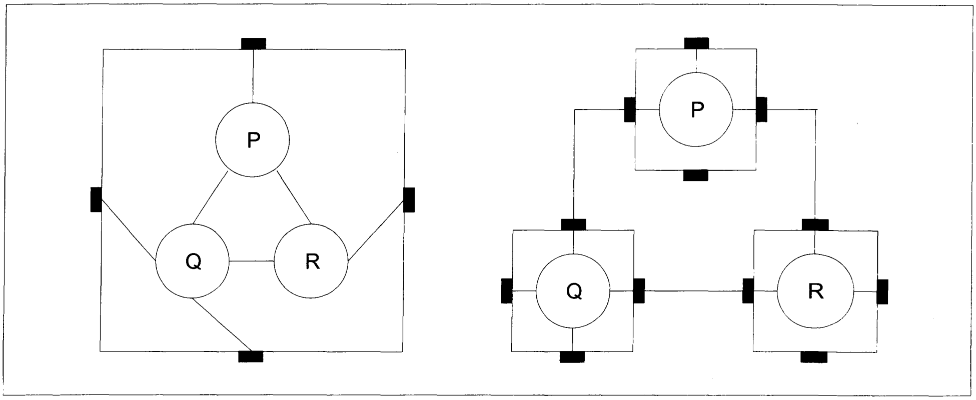


Figure 2.1 Mapping processes onto one or several transputers

The transputer and occam were designed together. All transputers include special instructions and hardware to provide maximum performance and optimal implementations of the occam model of concurrency and communications.

All transputer instruction sets are designed to enable simple, direct and efficient compilation of occam. Programming of I/O, interrupts and timing is standard on all transputers and conforms to the occam model.

Different transputer variants may have different instruction sets, depending on the desired balance of cost, performance, internal concurrency and special hardware. The occam level interface will, however, remain standard across all products.

2.2 occam overview

2.2.1 Processes

After it starts execution, a process performs a number of actions, and then either stops or terminates. Each action may be an assignment, an input, or an output. An assignment changes the value of a variable, an input receives a value from a channel, and an output sends a value to a channel.

At any time between its start and termination, a process may be ready to communicate on one or more of its channels. Each channel provides a one way connection between two concurrent processes; one of the processes may only output to the channel, and the other may only input from it.

Assignment

An assignment is indicated by the symbol `:=`. The example

```
v := e
```

sets the value of the variable `v` to the value of the expression `e` and then terminates, for example:

`x := 0` sets `x` to zero, and `x := x + 1` increases the value of `x` by 1.

Input

An input is indicated by the symbol `?` The example

```
c ? x
```

inputs a value from the channel `c`, assigns it to the variable `x` and then terminates.

Output

An output is indicated by the symbol ! The example

```
c ! e
```

outputs the value of the expression e to the channel c.

2.2.2 Constructions

A number of processes can be combined to form a construct. A construct is itself a process and can therefore be used as a component of another construct. Each component process of a construct is written two spaces further from the left hand margin, to indicate that it is part of the construct. There are four classes of constructs namely the sequential, parallel, conditional and the alternative construct.

Sequence

A sequential construct is represented by

```
SEQ
  P1
  P2
  P3
  ...
```

The component processes P1, P2, P3 ... are executed one after another. Each component process starts after the previous one terminates and the construct terminates after the last component process terminates. For example

```
SEQ
  c1 ?  x
  x  := x + 1
  c2 !  x
```

inputs a value, adds one to it, and then outputs the result.

Sequential constructs in occam are similar to programs written in conventional programming languages. Note, however, that they provide the performance and efficiency equivalent to that of an assembler for a conventional microprocessor.

Parallel

A parallel construct is represented by

```
PAR
  P1
  P2
  P3
  ...
```

The component processes P1, P2, P3 ... are executed together, and are called concurrent processes. The construct terminates after all of the component processes have terminated, for example:

```
PAR
  c1 ? x
  c2 ! y
```

allows the communications on channels c1 and c2 to take place together.

The parallel construct is unique to occam. It provides a straightforward way of writing programs which directly reflects the concurrency inherent in real systems. The implementation of parallelism on a single transputer is highly optimized so as to incur minimal process scheduling overhead.

Communication

Concurrent processes communicate only by using channels, and communication is synchronized. If a channel is used for input in one process, and output in another, communication takes place when both the inputting and the outputting processes are ready. The value to be output is copied from the outputting process to the inputting process, and the processes then proceed.

Communication between processes on a single transputer is via memory-to-memory data transfer. Between processes on different transputers it is via standard links. In either case the occam program is identical.

Conditional

A conditional construct

```
IF
  condition1
    P1
  condition2
    P2
  ...
```

means that `P1` is executed if `condition1` is true, otherwise `P2`is executed if `condition2` is true, and so on. Only one of the processes is executed, and then the construct terminates, for example:

```
IF
  x = 0
    y := y + 1
  x <> 0
    SKIP
```

increases `y` only if the value of `x` is 0.

Alternation

An alternative construct

```
ALT
  input1
    P1
  input2
    P2
  input3
    P3
  ...
```

waits until one of `input1`, `input2`, `input3` ... is ready. If `input1` first becomes ready, `input1` is performed, and then process `P1` is executed. Similarly, if `input2` first becomes ready, `input2` is performed, and then process `P2` is executed. Only one of the inputs is performed, then its corresponding process is executed and then the construct terminates, for example:

```
ALT
  count ? signal
    counter := counter + 1
  total ? signal
    SEQ
      out ! counter
      counter := 0
```

either inputs a signal from the channel `count`, and increases the variable `counter` by 1, or alternatively inputs from the channel `total`, outputs the current value of the counter, then resets it to zero.

The `ALT` construct provides a formal language method of handling external and internal events that must be handled by assembly level interrupt programming in conventional microprocessors.

Loop

```
WHILE condition
  P
```

repeatedly executes the process P until the value of the condition is false, for example:

```
WHILE (x - 5) > 0
  x := x - 5
```

leaves x holding the value of (x remainder 5) if x were positive.

Selection

A selection construct

```
CASE s
  n
    P1
  m,q
    P2
  ...
```

means that P1 is executed if s has the same value as n, otherwise P2 is executed if s has the same value as m or q, and so on, for example:

```
CASE direction
  up
    x := x + 1
  down
    x := x - 1
```

increases the value of x if direction is equal to up, otherwise if direction is equal to down the value of x is decreased.

Replication

A replicator is used with a SEQ, PAR, IF or ALT construction to replicate the component process a number of times. For example, a replicator can be used with SEQ to provide a conventional loop.

```
SEQ i = 0 FOR n
  P
```

causes the process P to be executed n times.

A replicator may be used with PAR to construct an array of concurrent processes.

```
PAR i = 0 FOR n
  Pi
```

constructs an array of n similar processes P0, P1, ..., Pn-1. The index i takes the values 0, 1, ..., n-1, in P0, P1, ..., Pn-1 respectively.

2.2.3 Types

Every variable, expression and value has a type, which may be a primitive type, array type, record type or variant type. The type defines the length and interpretation of data.

All implementations provide the primitive types shown in table 2.1.

CHAN OF *protocol*	Each communication channel provides communication between two concurrent processes. Each channel is of a type which allows communication of data according to the specified protocol.
TIMER	Each timer provides a clock which can be used by any number of concurrent processes.
BOOL	The values of type `BOOL` are true and false.
BYTE	The values of type `BYTE` are unsigned numbers `n` in the range 0 <=`n`< 256.
INT	Signed integers `n` in the range -2^{31} <=`n`< 2^{31}.
INT16	Signed integers `n` in the range -2^{15} <=`n`< 2^{15}.
INT32	Signed integers `n` in the range -2^{31} <=`n`< 2^{31}.
INT64	Signed integers `n` in the range -2^{63} <=`n`< 2^{63}.
REAL32	Floating point numbers stored using a sign bit, 8 bit exponent and 23 bit fraction in ANSI/IEEE Standard 754-1985 representation.[1]
REAL64	Floating point numbers stored using a sign bit, 11 bit exponent and 52 bit fraction in ANSI/IEEE Standard 754-1985 representation.[1]

Table 2.1 Types

2.2.4 Declarations, arrays and subscripts

A declaration `T x` declares `x` as a new channel, variable, timer or array of type `T`, for example:

```
INT x:
P
```

declares `x` as an integer variable for use in process `P`.

Array types are constructed from component types. For example `[ n ] T` is an array type constructed from `n` components of type `T`.

A component of an array may be selected by subscription, for example `v[e]` selects the `e`'th component of `v`.

A set of components of an array may be selected by subscription, for example `[v FROM e FOR c]` selects the `c` components `v[e], v[e + 1], ... v[e + c - 1]`. A set of components of an array may be assigned, input or output.

2.2.5 Procedures

A process may be given a name, for example:

```
PROC square (INT n)
  n := n * n
:
```

defines the procedure `square`. The name may be used as an instance of the process, for example:

```
square (x)
```

is equivalent to

```
n IS x:
n := n * n
```

[1] *IEEE Standard for Binary Floating-Point arithmetic* ANSI/IEEE Std 754-1985

2.2.6 Functions

A function can be defined in the same way as a procedure. For example:

```
INT FUNCTION factorial (VAL INT n)
  INT product:
  VALOF
    IF
      n >= 0
        SEQ
          product := 1
          SEQ i = 1 FOR n
            product := product * i
    RESULT product
:
```

defines the function `factorial`, which may appear in expressions such as

```
m := factorial (6)
```

2.2.7 Expressions

An expression is constructed from the operators given in table 2.2, from variables, numbers, the truth values `TRUE` and `FALSE`, and the brackets `(` and `)`.

Operator	Operand types	Description
`+ - * / REM`	integer, real	arithmetic operators
`PLUS MINUS TIMES AFTER`	integer	modulo arithmetic
`= <>`	any primitive	relational operators
`> < >= <=`	integer, real	relational operators
`AND OR NOT`	boolean	boolean operators
`/\ \/ >< ~`	integers	bitwise operators: and, or, xor, not
`<< >>`	integer	shift operators

Table 2.2 Operators

For example, the expression

```
(5 + 7) / 2
```

evaluates to 6, and the expression

```
(#1DF /\ #F0) >> 4
```

evaluates to `#D` (the character `#` introduces a hexadecimal constant).

A string is represented as a sequence of ASCII characters, enclosed in double quotation marks ″. If the string has `n` characters, then it is an array of type `[n]BYTE`.

2.2.8 Timer

All transputers incorporate a timer. The implementation directly supports the occam model of time. Each process can have its own independent timer, which can be used for internal measurement or for real time scheduling.

A timer input sets a variable to a value of type `INT` representing the time. The value is derived from a clock, which changes at regular intervals, for example:

```
tim ? v
```

sets the variable `v` to the current value of a free running clock, declared as the timer `tim`.

A delayed input takes the following form

```
        tim ? AFTER e
```

A delayed input is unable to proceed until the value of the timer satisfies (*timer* AFTER *e*). The comparison performed is a modulo comparison. This provides the effect that, starting at any point in the timer's cycle, the previous half cycle of the timer is considered as being before the current time, and the next half cycle is considered as being after the current time.

2.2.9 Peripheral access

The implementation of occam provides for peripheral access by extending the input and output primitives with a port input/output mechanism. A port is used like an occam channel, but has the effect of transferring information to and from a block of addresses associated with a peripheral.

Ports behave like occam channels in that only one process may input from a port, and only one process may output to a port. Thus ports provide a secure method of accessing external memory mapped status registers etc.

Note that there is no synchronization mechanism associated with port input and output. Any timing constraints which result from the use of asynchronous external hardware will have to be programmed explicitly. For example, a value read by a port input may depend upon the time at which the input was executed, and inputting at an invalid time would produce unusable data.

During applications development it is recommended that the peripheral is modelled by an occam process connected via channels.

2.3 Configuration

occam programs may be configured for execution on one or many transputers. The transputer development system provides the necessary tools for correctly distributing a program configured for many transputers.

Configuration does not affect the logical behavior of a program (see section four, Program development). However, it does enable the program to be arranged to ensure that performance requirements are met.

PLACED PAR

A parallel construct may be configured for a network of transputers by using the PLACED PAR construct. Each component process (termed a placement) is executed by a separate transputer. The variables and timers used in a placement must be declared within each placement process.

PRI PAR

On any individual transputer, the outermost parallel construct may be configured to prioritize its components. Each process is executed at a separate priority. The first process has the highest priority, the last process has the lowest priority. Lower priority components may only proceed when all higher priority components are unable to proceed.

2.3.1 INMOS standard links

Each link provides one channel in each direction between two transputers.

A channel (which must already have been declared) is associated with a link by a channel association, for example:

```
        PLACE Link0Input AT 4 :
```

3 Error handling

Errors in occam programs are either detected by the compiler or can be handled at runtime in one of three ways.

1. Cause the process to STOP allowing other processes to continue.
2. Cause the whole system to halt.
3. Have an arbitrary (undefined) effect.

The occam process STOP starts but never terminates. In method **1**, an errant process stops and in particular cannot communicate erroneous data to other processes. Other processes will continue to execute until they become dependent on data from the stopped process. It is therefore possible, for example, to write a process which uses a timeout to warn of a stopped process, or to construct a redundant system in which several processes performing the same task are used to enable the system to continue after one of them has failed.

Method **1** is the preferred method of executing a program.

Method **2** is useful for program development and can be used to bring transputers to an immediate halt, preventing execution of further instructions. The transputer **Error** output can be used to inform the transputer development system that such an error has occurred. No variable local to the process can be overwritten with erroneous data, facilitating analysis of the program and data which gave rise to the error.

Method **3** is useful only for optimizing programs which are known to be correct!

When a system has stopped or halted as a result of an error, the state of all transputers in the system can be analyzed using the transputer development system.

For languages other than occam, the transputer provides facilities for handling individual errors by software.

4 Program development

The development of programs for multiple processor systems can involve experimentation. In some cases, the most effective configuration is not always clear until a substantial amount of work has been done. For this reason, it is desirable that most of the design and programming can be completed before hardware construction is started.

4.1 Logical behavior

An important property of occam in this context is that it provides a clear notion of 'logical behavior'; this relates to those aspects of a program not affected by real time effects.

It is guaranteed that the logical behavior of a program is not altered by the way in which the processes are mapped onto processors, or by the speed of processing and communication. Consequently a program ultimately intended for a network of transputers can be compiled, executed and tested on a single computer used for program development.

Even if the application uses only a single transputer, the program can be designed as a set of concurrent processes which could run on a number of transputers. This design style follows the best traditions of structured programming; the processes operate completely independently on their own variables except where they explicitly interact, via channels. The set of concurrent processes can run on a single transputer or, for a higher performance product, the processes can be partitioned amongst a number of transputers.

It is necessary to ensure, on the development system, that the logical behavior satisfies the application requirements. The only ways in which one execution of a program can differ from another in functional terms result from dependencies upon input data and the selection of components of an ALT. Thus a simple method of ensuring that the application can be distributed to achieve any desired performance is to design the program to behave 'correctly' regardless of input data and ALT selection.

4.2 Performance measurement

Performance information is useful to gauge overall throughput of an application, and has to be considered carefully in applications with real time constraints.

Prior to running in the target environment, an occam program should be relatively mature, and indeed should be correct except for interactions which do not obey the occam synchronization rules. These are precisely the external interactions of the program where the world will not wait to communicate with an occam process which is not ready. Thus the set of interactions that need to be tested within the target environment are well identified.

Because, in occam, every program is a process, it is extremely easy to add monitor processes or simulation processes to represent parts of the real time environment, and then to simulate and monitor the anticipated real time interactions. The occam concept of time and its implementation in the transputer is important. Every process can have an independent timer enabling, for example, all the real time interactions to be modelled by separate processes and any time dependent features to be simulated.

4.3 Separate compilation of occam and other languages

A program portion which is separately compiled, and possibly written in a language other than occam, may be executed on a single transputer.

If the program is written in occam, then it takes the form of a single PROC, with only channel parameters. If the program is written in a language other than occam, then a run-time system is provided which provides input/output to occam channels.

Such separately compiled program portions are linked together by a framework of channels, termed a harness. The harness is written in occam. It includes all configuration information, and in particular specifies the transputer configuration in which the separately compiled program portion is executed.

Transputers are designed to allow efficient implementations of high level languages, such as C, Pascal and Fortran. Such languages will be available in addition to occam.

At runtime, a program written in such a language is treated as a single occam process. Facilities are provided in the implementations of these languages to allow such a program to communicate on occam channels. It can thus communicate with other such programs, or with programs written in occam. These programs may reside on the same transputer, in which case the channels are implemented in store, or may reside on different transputers, in which case the channels are implemented by transputer links.

It is therefore possible to implement occam processes in conventional high level languages, and arrange for them to communicate. It is possible for different parts of the same application to be implemented in different high level languages.

The standard input and output facilities provided within these languages are implemented by a well-defined protocol of communications on occam channels.

The development system provides facilities for management of separately compiled occam.

4.4 Memory map and placement

The low level memory model is of a signed address space.

Memory is byte addressed, the lowest addressed byte occupying the least significant byte position within the word.

The implementation of occam supports the allocation of the code and data areas of an occam process to specific areas of memory. Such a process must be a separately compiled PROC, and must not reference any variables and timers other than those declared within it.

5 Physical architecture

5.1 INMOS serial links

5.1.1 Overview

All transputers have several links. The link protocol and electrical characteristics form a standard for all INMOS transputer and peripheral products.

All transputers support a standard link communications frequency of 10 Mbits/sec. Some devices also support other data rates. Maintaining a standard communications frequency means that devices of mixed performance and type can intercommunicate easily.

Each link consists of two unidirectional signal wires carrying both data and control bits. The link signals are TTL compatible so that their range can be easily extended by inserting buffers.

The INMOS communication links provide for communication between devices on the same printed circuit board or between printed circuit boards via a back plane. They are intended to be used in electrically quiet environments in the same way as logic signals between TTL gates.

The number of links, and any communication speeds in addition to the standard speed of 10 Mbits/sec, are given in the **product data** for each product.

5.1.2 Link electrical specification

The quiescent state of the link signals is low, for a zero. The link input signals and output signals are standard TTL compatible signals.

For correct functioning of the links the specifications for maximum variation in clock frequency between two transputers joined by a link and maximum capacitive load must be met. Each transputer product also has specified the maximum permissible variation in delay in buffering, and minimum permissible edge gradients. Details of these specifications are provided in the product data.

Provided that these specifications are met then any buffering employed may introduce an arbitrary delay into a link signal without affecting its correct operation.

5.2 System services

5.2.1 Powering up and down, running and stopping

At all times the specification of input voltages with respect to the **GND** and **VCC** pins must be met. This includes the times when the **VCC** pins are ramping to 5 V, and also while they are ramping from 5 V down to 0 V.

The system services comprise the clocks, power, and signals used for initialization.

The specification includes minimum times that **VCC** must be within specification, the input clock must be oscillating, and the **Reset** signal must be high before **Reset** goes low. These specifications ensure that internal clocks and logic have settled before the transputer starts.

When the transputer is reset the memory interface is initialized (if present and configurable).

The processor and INMOS serial links start after reset. The transputer obeys a bootstrap program which can either be in off-chip ROM or can be received from one of the links. How to specify where the bootstrap program is taken from depends upon the type of transputer being used. The program will normally load up a larger program either from ROM or from a peripheral such as a disk.

During power down, as during power up, the input and output pins must remain within specification with respect to both **GND** and **VCC**.

A software error, such as arithmetic overflow, array bounds violation or divide by zero, causes an error flag to be set in the transputer processor. The flag is directly connected to the **Error** pin. Both the flag and the pin can be ignored, or the transputer stopped. Stopping the transputer on an error means that the error cannot cause further corruption.

As well as containing the error in this way it is possible to determine the state of the transputer and its memory at the time the error occurred.

5.2.2 Clock distribution

All transputers operate from a standard 5MHz input clock. High speed clocks are derived internally from the low frequency input to avoid the problems of distributing high frequency clocks. Within limits the mark-to-space ratio, the voltage levels and the transition times are immaterial. The limits on these are given in the product data for each product. The asynchronous data reception of the links means that differences in the clock phase between chips is unimportant.

The important characteristic of the transputer's input clock is its stability, such as is provided by a crystal oscillator. An R-C oscillator is inadequate. The edges of the clock should be monotonic (without kinks), and should not undershoot below -0.5 V.

5.3 Bootstrapping from ROM or from a link

The program which is executed after reset can either reside in ROM in the transputer's address space or it can be loaded via any one of the transputer's INMOS serial links.

The transputer bootstraps from ROM by transferring control to the top two bytes in memory, which will invariably contain a backward jump into ROM.

If bootstrapping from a link, the transputer bootstraps from the first link to receive a message. The first byte of the message is the count of the number of bytes of program which follow. The program is loaded into memory starting at a product dependent location *MemStart*, and then control is transferred to this address.

Messages subsequently arriving on other links are not acknowledged until the transputer processor obeys a process which inputs from them. The loading of a network of transputers is controlled by the transputer development system, which ensures that the first message each transputer receives is the bootstrap program.

5.4 Peripheral interfacing

All transputers contain one or more INMOS serial links. Certain transputer products also have other application specific interfaces. The peripheral control transputers contain specialized interfaces to control a specific peripheral or peripheral family.

In general, a transputer based application will comprise a number of transputers which communicate using INMOS links. There are three methods of communicating with peripherals.

The first is by employing peripheral control transputers (eg for graphics or disks), in which the transputer chip connects directly to the peripheral concerned (figure 5.1). The interface to the peripheral is implemented by special purpose hardware within the transputer. The application software in the transputer is implemented as an occam process, and controls the interface via occam channels linking the processor to the special purpose hardware.

The second method is by employing link adaptors (figure 5.2). These devices convert between a link and a specialized interface. The link adaptor is connected to the link of an appropriate transputer, which contains the application designer's peripheral device handler implemented as an occam process.

The third method is by memory mapping the peripheral onto the memory bus of a transputer (figure 5.3). The peripheral is controlled by memory accesses issued as a result of PORT inputs and outputs. The application designer's peripheral device handler provides a standard occam channel interface to the rest of the application.

The first transputers implement an event pin which provides a simple means for an external peripheral to request attention from a transputer.

In all three methods, the peripheral driver interfaces to the rest of the application via occam channels. Consequently, a peripheral device can be simulated by an occam process. This enables testing of all aspects of a transputer system before the construction of hardware.

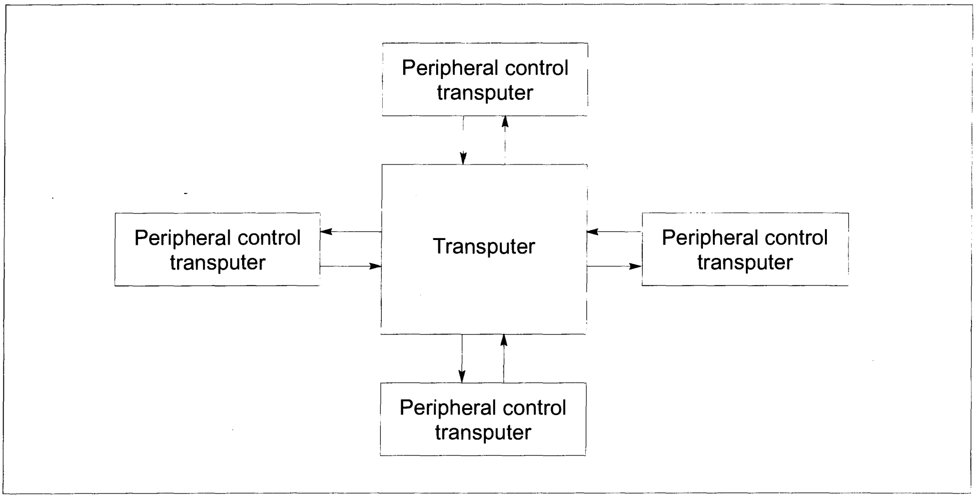

Peripheral control transputer
Peripheral control transputer
Transputer
Peripheral control transputer
Peripheral control transputer

Figure 5.1 Transputer with peripheral control transputers

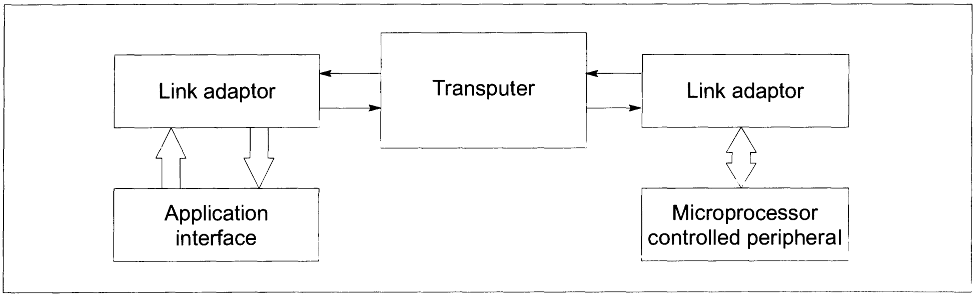


Figure 5.2 Transputer with link adaptors

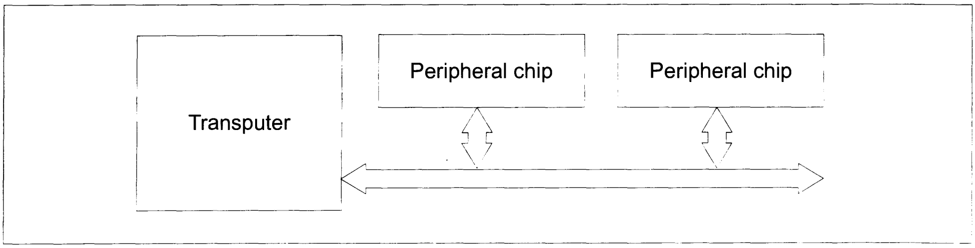


Figure 5.3 Memory mapped peripherals

Chapter 3

Transputer overview

1 Transputer internal architecture

Internally, a transputer consists of a memory, processor and communications system connected via a 32-bit bus. The bus also connects to the external memory interface, enabling additional local memory to be used. The processor, memory and communications system each occupy about 25% of the total silicon area, the remainder being used for power distribution, clock generators and external connections.

The processor contains instruction processing logic, instruction and work pointers, and an operand register. It directly accesses the high speed on-chip memory, which can store data or program. Where larger amounts of memory or programs in ROM are required, the processor has access to 4 Gbytes (32 bit) or 64Kbytes (16 bit) of memory via the External Memory Interface (EMI).

The floating point transputers each have an on-chip floating point unit. The small size and high performance of this unit come from a design which takes careful note of silicon economics. This contrasts starkly with conventional co-processors, where the floating point unit typically occupies more area than a complete micro-processor, and requires a second chip.

The block diagram figure 1.1 indicates the way in which the major blocks of the transputer are interconnected.

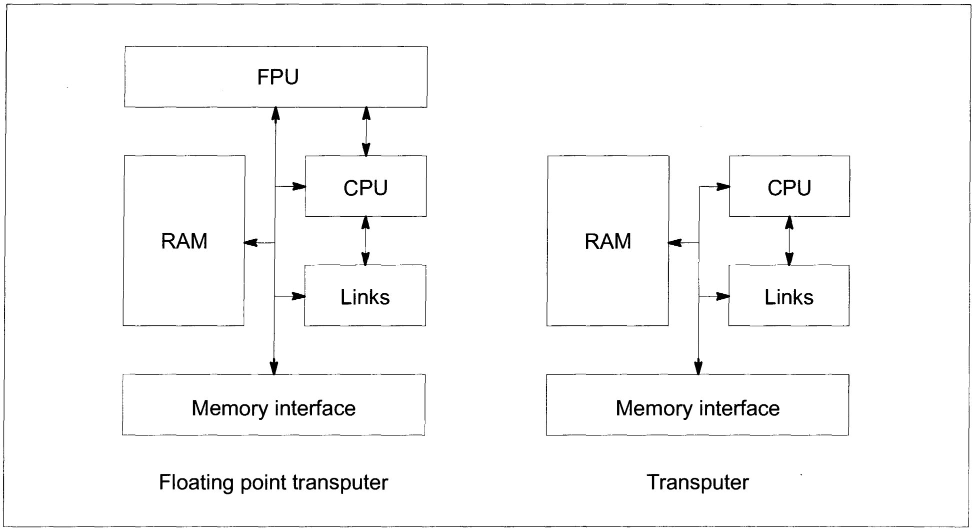


Figure 1.1 Transputer interconnections

The CPU of the transputer contains three registers (**A**, **B** and **C**) used for integer and address arithmetic, which form a hardware stack. Loading a value into the stack pushes **B** into **C**, and **A** into **B**, before loading **A**. Storing a value from **A** pops **B** into **A** and **C** into **B**. Similarly, the FPU includes a three register floating-point evaluation stack, containing the **AF**, **BF**, and **CF** registers. When values are loaded onto, or stored from the stack the **AF**, **BF** and **CF** registers push and pop in the same way as the **A**, **B** and **C** registers.

The addresses of floating point values are formed on the CPU stack, and values are transferred between the addressed memory locations and the FPU stack under the control of the CPU. As the CPU stack is used only to hold the addresses of floating point values, the wordlength of the CPU is independent of that of the FPU. Consequently, it would be possible to use the same FPU together with a 16-bit CPU. The transputer scheduler provides two priority levels. The FPU register stack is duplicated so that when the floating point transputer switches from low to high priority none of the state in the floating point unit is written to memory. This results in a worst-case interrupt response of about 3 μs. Furthermore, the duplication of the register stack enables floating point arithmetic to be used in an interrupt routine without any performance penalty.

1.1 Registers

The design of the transputer processor exploits the availability of fast on-chip memory by having only a small number of registers; the CPU contains six registers which are used in the execution of a sequential process. The small number of registers, together with the simplicity of the instruction set enables the processor to have relatively simple (and fast) data-paths and control logic.

The six registers are:

The workspace pointer which points to an area of store where local variables are kept.

The instruction pointer which points to the next instruction to be executed.

The operand register which is used in the formation of instruction operands.

The **A**, **B** and **C** registers which form an evaluation stack.

The **A**, **B** and **C** registers are the sources and destinations for most arithmetic and logical operations. Loading a value into the stack pushes **B** into **C**, and **A** into **B**, before loading **A**. Storing a value from **A**, pops **B** into **A** and **C** into **B**.

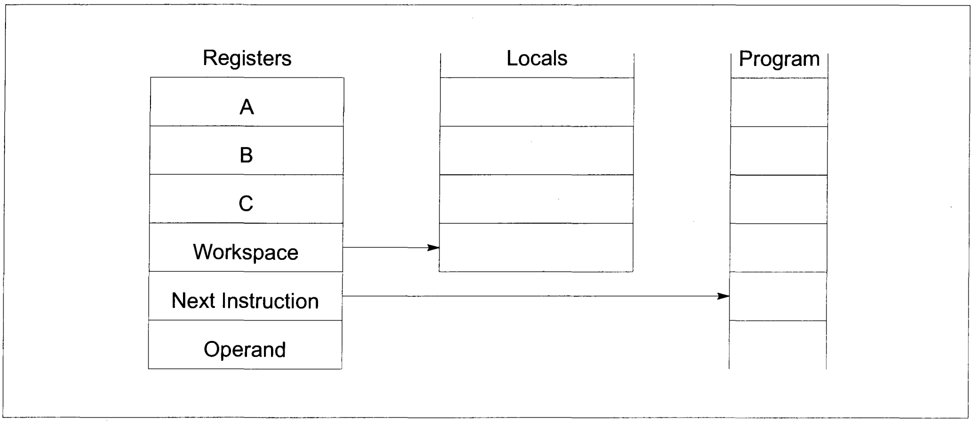


Figure 1.2 Registers used in sequential integer processes

Expressions are evaluated on the evaluation stack, and instructions refer to the stack implicitly. For example, the *add* instruction adds the top two values in the stack and places the result on the top of the stack. The use of a stack removes the need for instructions to respecify the location of their operands. Statistics gathered from a large number of programs show that three registers provide an effective balance between code compactness and implementation complexity.

No hardware mechanism is provided to detect that more than three values have been loaded onto the stack. It is easy for the compiler to ensure that this never happens.

Any location in memory can be accessed relative to the workpointer register, enabling the workspace to be of any size.

Further register details are given in *Transputer Instruction Set – A Compiler Writer's Guide*.

1.2 Instructions

It was a design decision that the transputer should be programmed in a high-level language. The instruction set has, therefore, been designed for simple and efficient compilation of high-level languages. It contains a relatively small number of instructions, all with the same format, chosen to give a compact representation of the operations most frequently occurring in programs. The instruction set is independent of the processor wordlength, allowing the same microcode to be used for transputers with different wordlengths. Each instruction consists of a single byte divided into two 4-bit parts. The four most significant bits of the byte are a function code, and the four least significant bits are a data value.

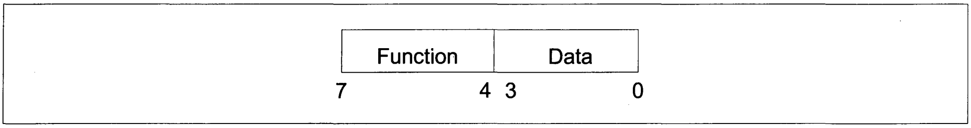


Figure 1.3 Instruction format

1.2.1 Direct functions

The representation provides for sixteen functions, each with a data value ranging from 0 to 15. Thirteen of these are used to encode the most important functions performed by any computer. These include:

load constant	*add constant*	
load local	*store local*	*load local pointer*
load non-local	*store non-local*	
jump	*conditional jump*	*call*

The most common operations in a program are the loading of small literal values, and the loading and storing of one of a small number of variables. The *load constant* instruction enables values between 0 and 15 to be loaded with a single byte instruction. The *load local* and *store local* instructions access locations in memory relative to the workspace pointer. The first 16 locations can be accessed using a single byte instruction.

The *load non-local* and *store non-local* instructions behave similarly, except that they access locations in memory relative to the **A** register. Compact sequences of these instructions allow efficient access to data structures, and provide for simple implementations of the static links or displays used in the implementation of high level programming languages such as occam, C, Fortran, Pascal, or ADA.

1.2.2 Prefix functions

Two more of the function codes are used to allow the operand of any instruction to be extended in length. These are:

prefix *negative prefix*

All instructions are executed by loading the four data bits into the least significant four bits of the operand register, which is then used as the instruction's operand. All instructions except the prefix instructions end by clearing the operand register, ready for the next instruction.

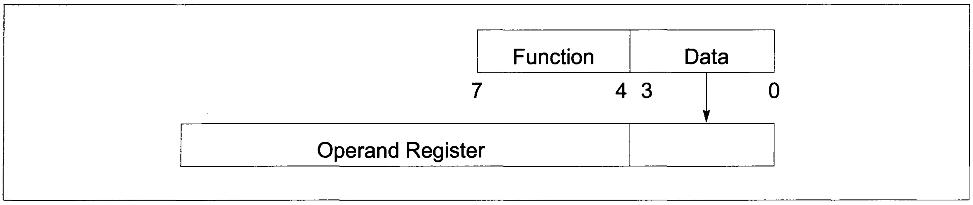


Figure 1.4 Instruction operand register

The *prefix* instruction loads its four data bits into the operand register, and then shifts the operand register up four places. The *negative prefix* instruction is similar, except that it complements the operand register before shifting it up. Consequently operands can be extended to any length up to the length of the operand register by a sequence of prefix instructions. In particular, operands in the range -256 to 255 can be represented using one prefix instruction.

The use of prefix instructions has certain beneficial consequences. Firstly, they are decoded and executed in the same way as every other instruction, which simplifies and speeds instruction decoding. Secondly, they simplify language compilation, by providing a completely uniform way of allowing any instruction to take an operand of any size. Thirdly, they allow operands to be represented in a form independent of the processor wordlength.

1.2.3 Indirect functions

The remaining function code, *operate*, causes its operand to be interpreted as an operation on the values held in the evaluation stack. This allows up to 16 such operations to be encoded in a single byte instruction. However, the prefix instructions can be used to extend the operand of an *operate* instruction just like any other. The instruction representation therefore provides for an indefinite number of operations.

The encoding of the indirect functions is chosen so that the most frequently occurring operations are represented without the use of a prefix instruction. These include arithmetic, logical and comparison operations such as

add *exclusive or* *greater than*

Less frequently occurring operations have encodings which require a single prefix operation (the transputer instruction set is not large enough to require more than 512 operations to be encoded!).

The IMS T800 has additional instructions which load into, operate on, and store from, the floating point register stack. It also contains new instructions which support color graphics, pattern recognition and the implementation of error correcting codes. These instructions have been added whilst retaining the existing instruction set. This has been possible because of the extensible instruction encoding used in transputers.

1.2.4 Expression evaluation

Evaluation of expressions sometimes requires use of temporary variables in the workspace, but the number of these can be minimised by careful choice of the evaluation order.

Program	Mnemonic	
x := 0	ldc	0
	stl	x
x := #24	pfix	2
	ldc	4
	stl	x
x := y + z	ldl	y
	ldl	z
	add	
	stl	x

Table 1.1 Expression evaluation

1.2.5 Efficiency of encoding

Measurements show that about 70% of executed instructions are encoded in a single byte (ie without the use of prefix instructions). Many of these instructions, such as *load constant* and *add* require just one processor cycle.

The instruction representation gives a more compact representation of high level language programs than more conventional instruction sets. Since a program requires less store to represent it, less of the memory bandwidth is taken up with fetching instructions. Furthermore, as memory is word accessed the processor will receive several instructions for every fetch.

Short instructions also improve the effectiveness of instruction prefetch, which in turn improves processor performance. There is an extra word of prefetch buffer so that the processor rarely has to wait for an instruction fetch before proceeding. Since the buffer is short, there is little time penalty when a jump instruction causes the buffer contents to be discarded.

1.3 Support for concurrency

The processor provides efficient support for the occam model of concurrency and communication. It has a microcoded scheduler which enables any number of concurrent processes to be executed together, sharing the processor time. This removes the need for a software kernel. The processor does not need to support the dynamic allocation of storage as the occam compiler is able to perform the allocation of space to concurrent processes.

A process starts, performs a number of actions, and then either stops without completing or terminates complete. Typically, a process is a sequence of instructions. A transputer can run several processes in parallel (concurrently). Processes may be assigned either high or low priority, and there may be any number of each.

At any time, a concurrent process may be

active	-	being executed
	-	on a list waiting to be executed
inactive	-	ready to input
	-	ready to output
	-	waiting until a specified time

The scheduler operates in such a way that inactive processes do not consume any processor time. The active processes waiting to be executed are held on a list. This is a linked list of process workspaces, implemented using two registers, one of which points to the first process on the list, the other to the last. In figure 1.5, S is executing, and P, Q and R are active, awaiting execution. Only the low priority process queue registers are shown; the high priority process ones perform in a similar manner.

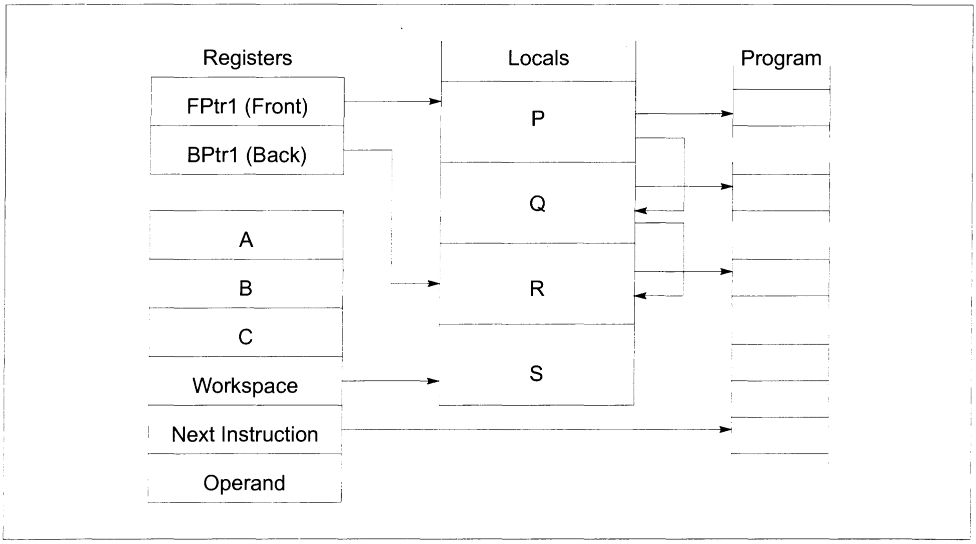


Figure 1.5 Linked process list

Function	High Priority	Low Priority
Pointer to front of active process list	**Fptr0**	**Fptr1**
Pointer to back of active process list	**Bptr0**	**Bptr1**

Table 1.2 Priority queue control registers

A process is executed until it is unable to proceed because it is waiting to input or output, or waiting for the timer. Whenever a process is unable to proceed, its instruction pointer is saved in its workspace and the next process is taken from the list. Actual process switch times are very small as little state needs to be saved; it is not necessary to save the evaluation stack on rescheduling.

In order for several processes to operate in parallel, a low priority process is only permitted to run for a maximum of two time slices before it is forcibly descheduled at the next descheduling point (section 2). The time slice period is 5120 cycles of the external 5 MHz clock, giving ticks approximately 1 ms apart.

A process can only be descheduled on certain instructions, known as descheduling points. As a result, an expression evaluation can be guaranteed to execute without the process being timesliced part way through.

Whenever a process is unable to proceed, its instruction pointer is saved in the process workspace and the next process taken from the list. Process scheduling pointers are updated by instructions which cause scheduling operations, and should not be altered directly. Actual process switch times are less than 1 μs, as little state needs to be saved and it is not necessary to save the evaluation stack on rescheduling.

The processor provides a number of special operations to support the process model. These include:

start process *end process*

When a parallel construct is executed, *start process* instructions are used to create the necessary concurrent processes. A *start process* instruction creates a new process by adding a new workspace to the end of the scheduling list, enabling the new concurrent process to be executed together with the ones already being executed.

The correct termination of a parallel construct is assured by use of the *end process* instruction. This uses a workspace location as a counter of the parallel construct components which have still to terminate. The counter is initialised to the number of components before the processes are *started*. Each component ends with an *end process* instruction which decrements and tests the counter. For all but the last component, the counter is non zero and the component is descheduled. For the last component, the counter is zero and the main process continues.

1.3.1 Priority

The transputer supports two levels of priority. Priority 1 (low priority) processes are executed whenever there are no active priority 0 (high priority) processes.

High priority processes are expected to execute for a short time. If one or more high priority processes are able to proceed, then one is selected and runs until it has to wait for a communication, a timer input, or until it completes processing.

If no process at high priority is able to proceed, but one or more processes at low priority are able to proceed, then one is selected.

Low priority processes are periodically timesliced to provide an even distribution of processor time between computationally intensive tasks.

If there are **n** low priority processes, then the maximum latency from the time at which a low priority process becomes active to the time when it starts processing is 2**n**-2 timeslice periods. It is then able to execute for between one and two timeslice periods, less any time taken by high priority processes. This assumes that no process monopolises the transputer's time; i.e. it has a distribution of descheduling points (section 2).

Each timeslice period lasts for 5120 cycles of the external 5 MHz input clock (approximately 1 ms at the standard frequency of 5 MHz).

1.3.2 Interrupt latency

If a high priority process is waiting for an external channel to become ready, and if no other high priority process is active, then the interrupt latency for the IMS T400, IMS T425, and IMS T426 (from when the channel becomes ready to when the process starts executing) is typically 19 processor cycles, a maximum of 58 cycles (assuming use of on-chip RAM).

For the IMS T801 and IMS T805 the interrupt latency is typically 19 processor cycles, a maximum of 78 cycles (assuming use of on-chip RAM). If the floating point unit is not being used at the time then the maximum interrupt latency is only 58 cycles. To ensure this latency, certain instructions are interruptable.

For the IMS T222 and IMS T225 the interrupt latency is typically 19 processor cycles, a maximum of 53 cycles (assuming use of on-chip RAM).

1.4 Communications

Communication between processes is achieved by means of channels. occam communication is point-to-point, synchronized and unbuffered. As a result, a channel needs no process queue, no message queue and no message buffer.

A channel between two processes executing on the same transputer is implemented by a single word in memory; a channel between processes executing on different transputers is implemented by point-to-point links. The processor provides a number of operations to support message passing, the most important being

input message *output message*

The *input message* and *output message* instructions use the address of the channel to determine whether the channel is internal or external. This means that the same instruction sequence can be used for both hard and soft channels, allowing a process to be written and compiled without knowledge of where its channels are connected.

As in the occam model, communication takes place when both the inputting and outputting processes are ready. Consequently, the process which first becomes ready must wait until the second one is also ready.

A process performs an input or output by loading the evaluation stack with a pointer to a message, the address of a channel, and a count of the number of bytes to be transferred, and then executing an *input message* or an *output message* instruction.

1.4.1 Internal channel communication

At any time, an internal channel (a single word in memory) either holds the identity of a process, or holds the special value *empty*. The channel is initialized to *empty* before it is used.

When a message is passed using the channel, the identity of the first process to become ready is stored in the channel, and the processor starts to execute the next process from the scheduling list. When the second process to use the channel becomes ready, the message is copied, the waiting process is added to the scheduling list, and the channel reset to its initial state. It does not matter whether the inputting or the outputting process becomes ready first.

In figure 1.6, a process P is about to execute an output instruction on an ‘empty’ channel C. The evaluation stack holds a pointer to a message, the address of channel C, and a count of the number of bytes in the message.

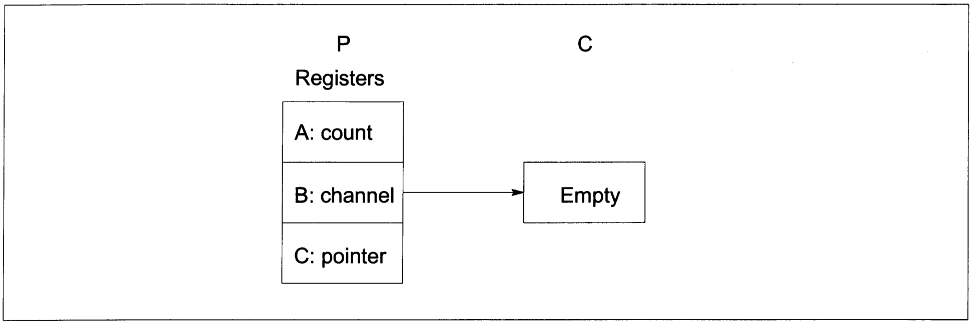


Figure 1.6 Output to empty channel

After executing the output instruction, the channel C holds the address of the workspace of P, and the address of the message to be transferred is stored in the workspace of P. P is descheduled, and the process starts to execute the next process from the scheduling list.

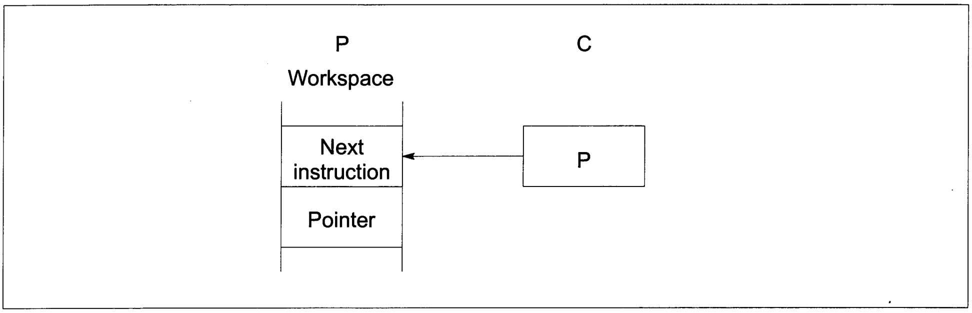


Figure 1.7

The channel C and the process P remain in this state until a second process, Q executes an output instruction on the channel.

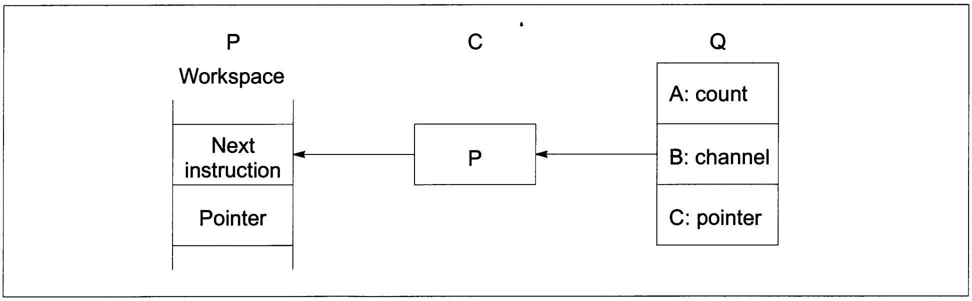


Figure 1.8

The message is copied, the waiting process P is added to the scheduling list, and the channel C is reset to its initial 'empty' state.

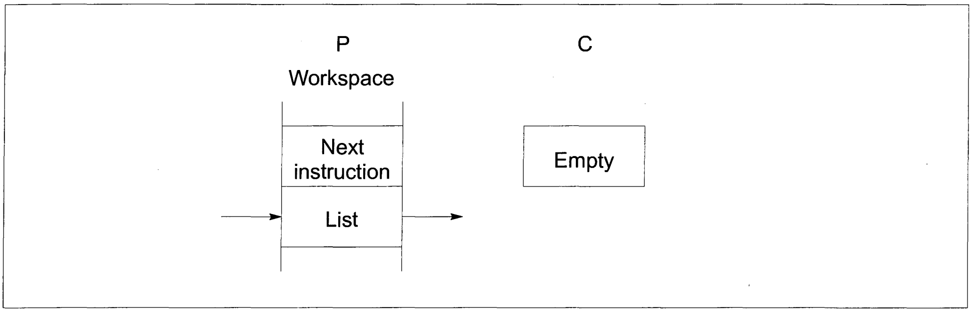


Figure 1.9

1.4.2 External channel communication

When a message is passed via an external channel the processor delegates to an autonomous link interface the job of transferring the message and deschedules the process. When the message has been transferred the link interface causes the processor to reschedule the waiting process. This allows the processor to continue the execution of other processes whilst the external message transfer is taking place.

Each link interface uses three registers:

a pointer to a process workspace

a pointer to a message

a count of bytes in the message

In figure 1.10 processes P and Q executed by different transputers communicate using a channel C implemented by a link connecting two transputers. P outputs, and Q inputs.

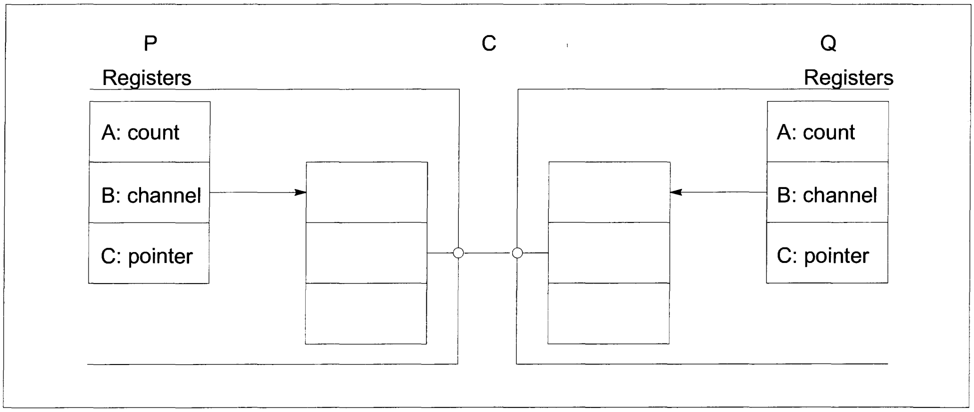


Figure 1.10 Communication between transputers

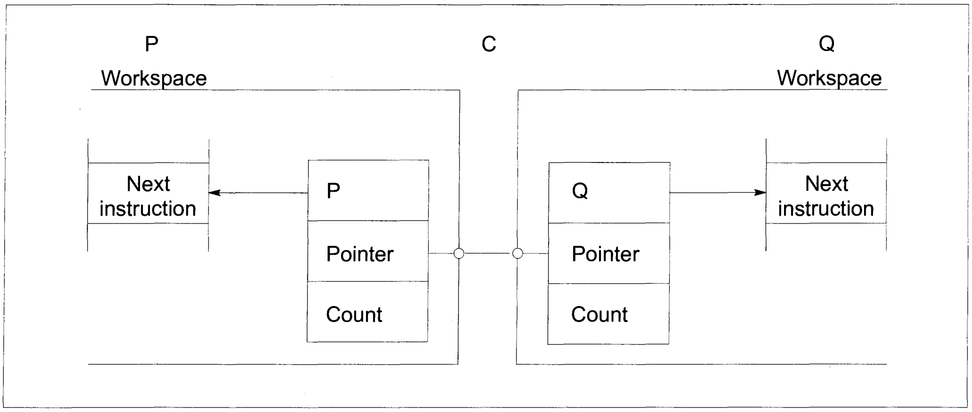


Figure 1.11

When P executes its output instruction, the registers in the link interface of the transputer executing P are initialized, and P is descheduled. Similarly, when Q executes its input instruction, the registers in the link interface of the process executing Q are initialized, and Q is descheduled (figure 1.11).

The message is now copied through the link, after which the workspaces of P and Q are returned to the corresponding scheduling lists (figure 1.12). The protocol used on P and Q ensures that it does not matter which of P and Q first becomes ready.

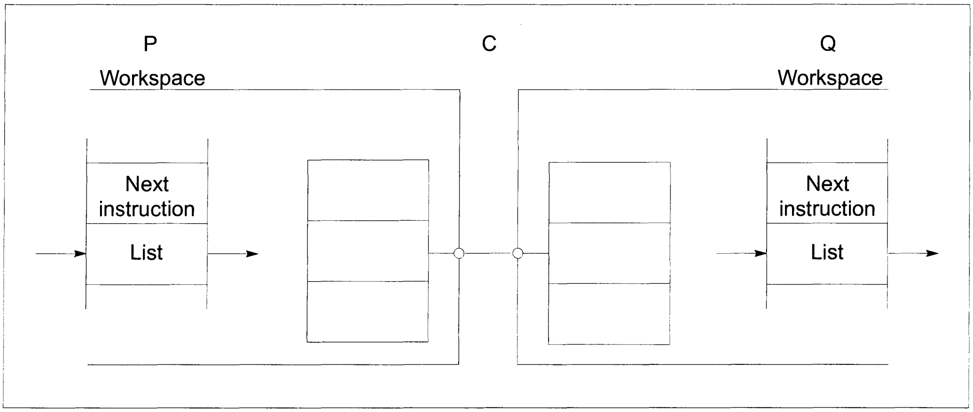


Figure 1.12

1.4.3 Communication links

A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer by two one-directional signal wires, along which data is transmitted serially. The two wires provide two occam channels, one in each direction. This requires a simple protocol to multiplex data and control information. Messages are transmitted as a sequence of bytes, each of which must be acknowledged before the next is transmitted. A byte of data is transmitted as a start bit followed by a one bit followed by eight bits of data followed by a stop bit. An acknowledgement is transmitted as a start bit followed by a stop bit. An acknowledgement indicates both that a process was able to receive the data byte and that it is able to buffer another byte.

The protocol permits an acknowledgement to be generated as soon as the receiver has identified a data packet. In this way the acknowledgement can be received by the transmitter before all of the data packet has been transmitted and the transmitter can transmit the next data packet immediately. Some transputers do not implement this overlapping and achieve a data rate of 0.8 Mbytes/sec using a link to transfer data in one direction. However, by implementing the overlapping and including sufficient buffering in the link hardware, the rate can be more than doubled to achieve 1.8 Mbytes/sec in one direction, and 2.4 Mbytes/sec when the link carries data in both directions. The diagram below shows the signals that would be observed on the two link wires when a data packet is overlapped with an acknowledgement.

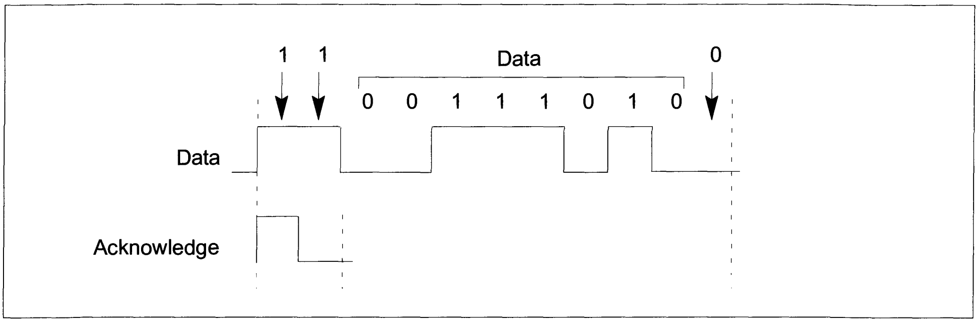


Figure 1.13 OS link data and acknowledge formats

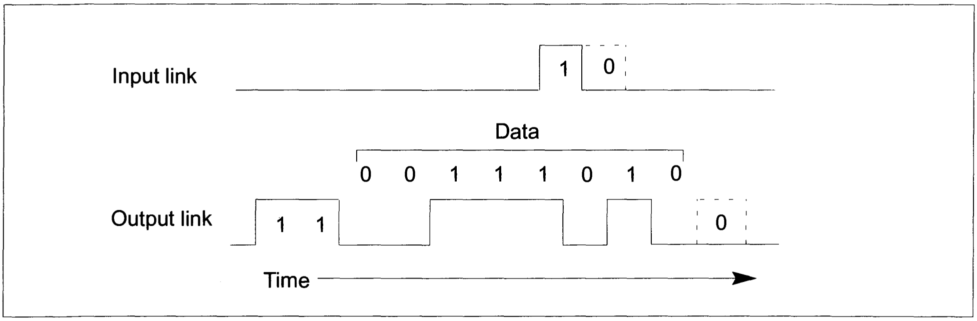


Figure 1.14 OS link data and acknowledge formats

1.5 Timer

The transputer has two 32 bit (IMS T801, IMS T805, IMS T400, IMS T425, IMS T426) or 16 bit (IMS T222, IMS T225) timer clocks which 'tick' periodically. The timers provide accurate process timing, allowing processes to deschedule themselves until a specific time.

IMS T222, IMS T225 — One timer is accessible only to high priority processes and is incremented every microsecond, cycling completely in approximately 65 milliseconds. The other is accessible only to low priority processes and is incremented every 64 microseconds, giving exactly 15625 ticks in one second. It has a full period of approximately four seconds.

IMS T801, IMS T805, IMS T400, IMS T425, IMS T426 — One timer is accessible only to high priority processes and is incremented every microsecond, cycling completely in approximately 4295 seconds. The other is accessible only to low priority processes and is incremented every 64 microseconds, giving exactly 15625 ticks in one second. It has a full period of approximately 76 hours.

Clock0	Current value of high priority (level 0) process clock
Clock1	Current value of low priority (level 1) process clock
TNextReg0	Indicates time of earliest event on high priority (level 0) timer queue
TNextReg1	Indicates time of earliest event on low priority (level 1) timer queue

Table 1.3 Timer registers

The current value of the processor clock can be read by executing a *load timer* instruction. A process can arrange to perform a *timer input*, in which case it will become ready to execute after a specified time has been reached. The *timer input* instruction requires a time to be specified. If this time is in the 'past' (i.e. *ClockReg* AFTER *SpecifiedTime*) then the instruction has no effect. If the time is in the 'future' (i.e. *SpecifiedTime* AFTER *Clockreg* or *SpecifiedTime* = *ClockReg*) then the process is descheduled. When the specified time is reached the process is scheduled again.

Figure 1.15 shows two processes waiting on the timer queue, one waiting for time 21, the other for time 31.

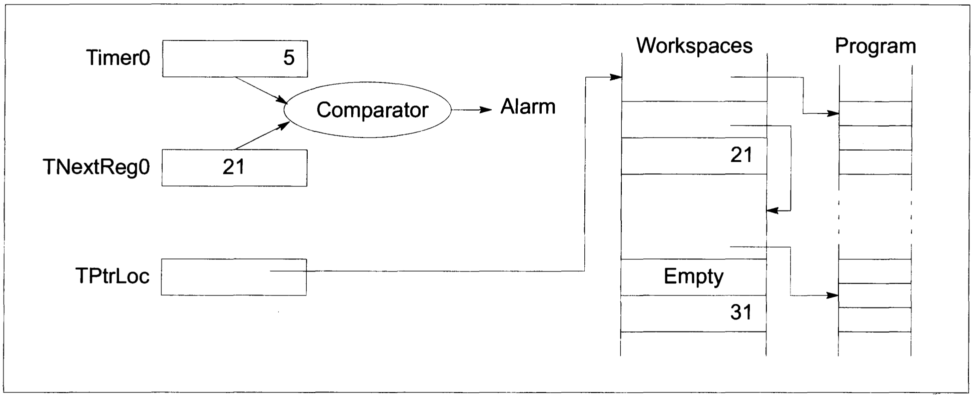


Figure 1.15 Timer registers

1.6 Alternative

The occam alternative construct enables a process to wait for input from any one of a number of channels, or until a specific time occurs. This requires special instructions, as the normal *input* instruction deschedules a process until a specific channel becomes ready, or until a specific time is reached. The instructions are:

enable channel	*disable channel*
enable timer	*disable timer*
alternative wait	

The alternative is implemented by 'enabling' the channel input or timer input specified in each of its components. The 'alternative wait' is then used to deschedule the process if none of the channel or timer inputs is ready; the process will be re-scheduled when any one of them becomes ready. The channel and timer inputs are then 'disabled'. The 'disable' instructions are also designed to select the component of the alternative to be executed; the first component found to be ready is executed.

1.7 Floating point instructions

The core of the floating point instruction set was established fairly early in the design of the floating point transputer. This core includes simple load, store and arithmetic instructions. Examination of statistics derived from FORTRAN programs suggested that the addition of some more complex instructions would improve performance and code density. Proposed changes to the instruction set were assessed by examining their effect on a number of numerical programs. For each proposed instruction set, a compiler was constructed, the programs compiled with it, and the resulting code then run on a simulator. The resulting instruction set is now described.

In the floating point transputer operands are transferred between the transputer's memory and the floating point evaluation stack by means of floating point load and store instructions. There are two groups of such instructions, one for single length numbers, one for double length. In the description of the load and store instructions which follow only the double length instructions are described. However, there are single length instructions which correspond with each of the double length instructions.

The address of a floating point operand is computed on the CPU's stack and the operand is then loaded, from the addressed memory location, onto the FPU's stack. Operands in the floating point stack are tagged with their length. The operand's tag will be set when the operand is loaded or is computed. The tags allow the number of instructions needed for floating point operations to be reduced; there is no need, for example, to have both *floating add single* and *floating add double* instructions; a single *floating add* will suffice.

1.7.1 Optimizing use of the stack

The depth of the register stacks in the CPU and FPU is carefully chosen. Floating point expressions commonly have embedded address calculations, as the operands of floating point operators are often elements of one dimensional or two dimensional arrays. The CPU stack is deep enough to allow most integer calculations and address calculations to be performed within it. Similarly, the depth of the FPU stack allows most floating point expressions to be evaluated within it, employing the CPU stack to form addresses for the operands.

No hardware is used to deal with stack overflow. A compiler can easily examine expressions and introduce temporary variables in memory to avoid stack overflow. The number of such temporary variables can be minimized by careful choice of the evaluation order; an algorithm to perform this optimization is given in the Prentice Hall publication *Transputer Instruction Set – A Compiler Writer's Guide*. The algorithm is used to optimize the use of the integer stack of the transputer CPU.

1.7.2 Concurrent operation of FPU and CPU

In the floating point transputer the FPU operates concurrently with the CPU. This means that it is possible to perform an address calculation in the CPU whilst the FPU performs a floating point calculation. This can lead to significant performance improvements in real applications which access arrays heavily. This aspect of the floating point transputer's performance was carefully assessed, partly through examination of the 'Livermore Loops' (refer to *The Livermore Fortran Kernels: A Computer Test of the Numerical Performance Range*). These are a collection of small kernels designed to represent the types of calculation performed on super-computers. They are of interest because they contain constructs which occur in real programs which are not represented in such programs as the Whetstone benchmark. In particular, they contain accesses to two and three-dimensional arrays, operations where the concurrency within the floating point transputer is used to good effect. In some cases the compiler is able to choose the order of performing address calculations so as to maximize overlapping; this involves a modification of the algorithm mentioned earlier.

As a simple example of overlapping consider the implementation of Livermore Loop 7. The occam program for loop 7 is as follows:

```
-- LIVERMORE LOOP 7
SEQ k = 0 FOR n
  x[k] :=   u[k] + ((( r*(z[k] + (r*y[k]))) +
           (t*((u[k+3] + (r*(u[k+2] + (r*u[k+1])))))) +
           (t*((u[k+6] + (r*(u[k+5] + (r*u[k+4])))))))
```

The first stage in the computation of this is to load the value `y[k]`. This requires a sequence of four instructions. A further three instructions cause `r` to be loaded and the FPU multiply to be initiated.

Although the floating point multiplication takes several cycles to complete, the CPU is able to continue executing instructions whilst the FPU performs the multiplication. Thus the CPU can execute the next segment of code which computes the address of `z[k]` whilst the FPU performs the multiplication.

Finally, the value `z[k]` is pushed onto the floating point stack and added to the previously computed subexpression `r*y[k]`. It is not until value `z[k]` is loaded that the CPU needs to synchronize with the FPU.

The computation of the remainder of the expression proceeds in the same way, and the FPU never has to wait for the CPU to perform an address calculation.

1.8 Floating point unit design

In designing a concurrent systems component such as a transputer, it is important to maximize the performance obtained from a given area of silicon; many components can be used together to deliver more performance. This contrasts with the design of a conventional co-processor where the aim is to maximize the performance of a single processor by the use of a large area of silicon. As a result, in designing the floating point transputer, the performance benefits of silicon hungry devices such as barrel shifters and flash multipliers were carefully examined.

A flash multiplier is too large to fit on chip together with the processor, and would therefore necessitate the use of a separate co-processor chip. The introduction of a co-processor interface to a separate chip slows down the rate at which operands can be transferred to and from the floating point unit. Higher performance can, therefore, be obtained from a slow multiplier on the same chip as the processor than from a fast one on a separate chip. This leads to an important conclusion: *a separate co-processor chip is not appropriate for scalar floating point arithmetic.* A separate co-processor would be effective where a large amount of work can be handed to the co-processor by transferring a small amount of information; for example a vector co-processor would require only the addresses of its vector operands to be transferred via the co-processor interface.

It turns out that a flash multiplier also operates much more quickly than is necessary. Only a pipelined vector processor can deliver operands at a rate consistent with the use of such devices. In fact, any useful floating point calculation involves more operand accesses than operations. As an example consider the assignment `y[i] := y[i] + (t * x[i])` which constitutes the core of the LINPACK floating point benchmark. To perform this it is necessary to load three operands, perform two operations and to store a result. If we assume that it takes twice as long to perform a floating point operation as to load or store a floating point number then the execution time of this example would be evenly split between operand access time and operation time. This means that there would be at most a factor of two available in performance improvement from the use of an infinitely fast floating point unit!

Unlike a flash multiplier, a fast normalizing shifter is important for fast floating point operation. When implementing IEEE arithmetic it may be necessary to perform a long shift on every floating point operation and unless a fast shifter is incorporated into the floating point unit the maximum operation time can become very long. Fortunately, unlike a flash multiplier, it is possible to design a fast shifter in a reasonable area of silicon. The shifter used is designed to perform a shift in a single cycle and to normalize in two cycles.

Consequently, the floating point unit contains a fast normalizing shifter but not a flash multiplier. However there is a certain amount of logic devoted to multiplication and division. Multiplication is performed three-bits per cycle, and division is performed two-bits per cycle. Figure 1.16 illustrates the physical layout of the floating point unit.

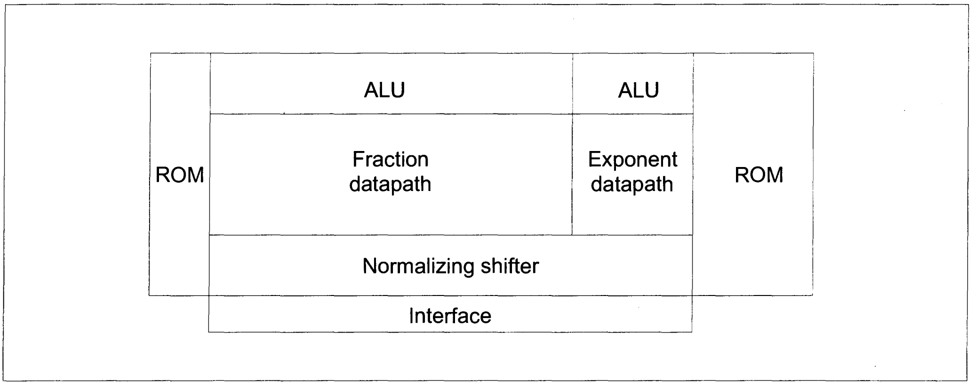


Figure 1.16 Floating point unit block diagram

The datapaths contain registers and shift paths. The fraction datapath is 59 bits wide, and the exponent data path is 13 bits wide. The normalizing shifter interfaces to both the fraction data path and the exponent datapath. This is because the data to be shifted will come from the fraction datapath whilst the magnitude of the shift is associated with the exponent datapath. One further interesting aspect of the design is the microcode ROM. Although the diagram shows two ROMs, they are both part of the same logical ROM. This has been split in two so that control signals do not need to be bussed through the datapaths.

1.9 Graphics capability

The fast block move instructions of the transputers make them suitable for use in graphics applications using byte-per-pixel color displays. The block move on the transputer is designed to saturate the memory bandwidth, moving any number of bytes from any byte boundary in memory to any other byte boundary using the smallest possible number of word read and write operations.

Some transputers extend this capability by incorporation of a two-dimensional version of the block move (`Move2d`) which can move windows around a screen at full memory bandwidth, and conditional versions of the same block move which can be used to place templates and text into windows. One of these operations (`Draw2d`) copies bytes from source to destination, writing only non-zero bytes to the destination. A new object of any shape can therefore be drawn on top of the current image. A further operation (`Clip2d`) copies only zero bytes in the source. All of these instructions achieve the speed of the simple move instruction, enabling a 1 million pixel screen to be drawn many times per second. Unlike the conventional 'bit-blt' instruction, it is never necessary to read the destination data.

1.9.1 Example – drawing colored text

Drawing proportional spaced text provides a simple example of the use of the two-dimensional move instructions. The font is stored in a two dimensional array `Font`; the height of `Font` is the fixed character height, and the start of each character is defined by an array `start`. The textures of the character and its background are selected from an array of textures; the textures providing a range of colors or even stripes and tartans!

An occam procedure to perform such drawing is given below and the effect of each stage in the drawing process is illustrated by the diagrams on the final page of this document. First, (1) the texture for the character is selected and copied to a temporary area and (2) the character in the font is used to clip this texture to the appropriate shape. Then (3) the background texture is selected and copied to the screen, and (4) the new character drawn on top of it.

```
-- Draw character ch in texture F on background texture B
 PROC DrawChar(VAL INT Ch, F, B)
   SEQ
```

```
IF
  (x + width[ch]) > screenwidth
    SEQ
      x := 0
      y := y + height
  (x + width[ch]) <= screenwidth
    SKIP
[height] [maxwidth] BYTE Temp :
SEQ
  Move2d(Texture[F],0,0, Temp,0,0, width[ch],height)
  Clip2d(Font[ch],start[ch],0, Temp,0,0, width[ch],height)
  Move2d(Texture[B],0,0, Screen,x,y, width[ch],height))
  Draw2d(Temp,0,0, Screen,x,y, width[ch],height)
  x := x + width[ch]
```

This procedure will fill a 1 million pixel screen with proportionally spaced characters in about 1/6 second. Obviously, a simpler and faster version could be used if the character color or background color was restricted. The operation of this procedure is illustrated in figure 1.17.

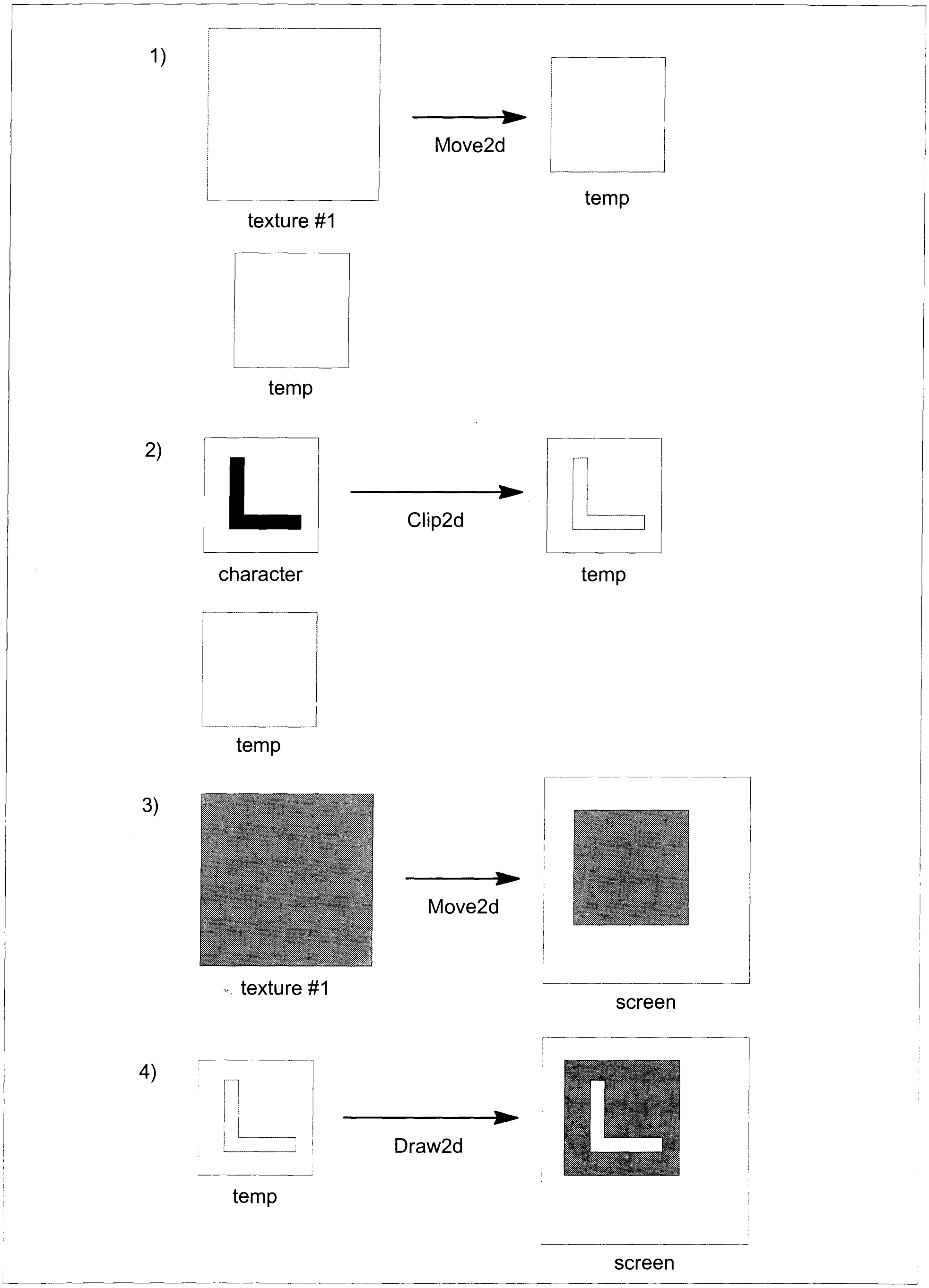


Figure 1.17 Use of enhanced graphics instructions

2 Conclusion

The INMOS transputer family is a range of system components which can be used to construct high performance concurrent systems. As all members of the family incorporate INMOS communications links, a system may be constructed from different members of the family. All transputers provide hardware support for concurrency and offer exceptional performance on process scheduling, inter-process communication and inter-transputer communication.

The design of the transputers takes careful note of silicon economics. The central processor used in the transputer offers a performance comparable with that of other VLSI processors several times larger. The small size of the processor allows a memory and communications system to be integrated on to the same VLSI device. This level of integration allows very fast access to memory and very fast inter-transputer communication. Similarly, the transputer floating point unit is integrated into the same device as the central processor, eliminating the delays inherent in communicating data between devices.

Transputer instruction set summary

1 Introduction

The Function Codes table 7.1 (page 50) gives the basic function code set. Where the operand value is less than 16, a single byte encodes the complete instruction. If the operand value is greater than 15, one prefix instruction (*pfix*) is required for each additional four bits of the operand. If the operand is negative the first prefix instruction will be *nfix*. Examples of prefix coding are given in table 1.1.

Mnemonic		Function code	Memory code
ldc	#3	#4	#43
ldc	#35		
is coded as			
pfix	#3	#2	#23
ldc	#5	#4	#45
ldc	#987		
is coded as			
pfix	#9	#2	#29
pfix	#8	#2	#28
ldc	#7	#4	#47
ldc	–31	(*ldc* #FFFFFFE1) (*ldc* #FFE1) †	
is coded as			
nfix	#1	#6	#61
ldc	#1	#4	#41
† IMS T222, IMS T225			

Table 1.1 *prefix* coding

Tables 7.2 to 8.7 (pages 51–58) give details of the operation codes. Where an operation code is less than 16 (e.g. *add*: operation code **05**), the operation can be stored as a single byte comprising the *operate* function code **F** and the operand (**5** in the example). Where an operation code is greater than 15 (e.g. *ladd*: operation code **16**), the *prefix* function code **2** is used to extend the instruction.

Mnemonic		Function code	Memory code
add	*(op. code #5)*		#F5
is coded as			
opr	add	#F	#F5
ladd	*(op. code #16)*		#21F6
is coded as			
pfix	#1	#2	#21
opr	#6	#F	#F6

Table 1.2 *operate* coding

Product identity numbers

The load device identity (*lddevid*) instruction (table 7.13) pushes the device type identity into the A register. Each product is allocated a unique group of numbers for use with the *lddevid* instruction. There is no *lddevid* on the IMS T222. Product identity numbers are given in table 1.3.

Product	Identity numbers
IMS T425	0 to 9 inclusive
IMS T805	10 to 19 inclusive
IMS T801	20 to 29 inclusive
IMS T426	30 to 39 inclusive
IMS T225	40 to 49 inclusive
IMS T400	50 to 59 inclusive

Table 1.3 Product identity numbers

Floating point unit

In the floating point unit (FPU) basic addition, subtraction, multiplication and division operations are performed by single instructions. However, certain less frequently used floating point instructions are selected by a value in register *A* (when allocating registers, this should be taken into account). A *load constant* instruction *ldc* is used to load register *A*; the *floating point entry* instruction *fpentry* then uses this value to select the floating point operation. This pair of instructions is termed a *selector sequence.*

In the Floating Point Operation Codes tables 8.1 to 8.7, a selector sequence code is indicated in the Memory Code column by **s**. The code given in the Operation Code column is the indirection code, the operand for the *ldc* instruction.

The FPU and processor operate concurrently, so the actual throughput of floating point instructions is better than that implied by simply adding up the instruction times. For full details see *Transputer Instruction Set – A Compiler Writer's Guide.*

Notation

The Processor Cycles column refers to the number of periods **TPCLPCL** (refer to **ProcClockOut** section of specific device e.g. IMS T805 page 88) taken by an instruction executing in internal memory. The number of cycles is given for the basic operation only; where the memory code for an instruction is two bytes, the time for the *prefix* function (one cycle) should be added. For a 20 MHz transputer one cycle is 50 ns. Some instruction times vary. Where a letter is included in the cycles column it is interpreted from table 1.4.

Ident	Interpretation
b	Bit number of the highest bit set in register **A**. Bit 0 is the least significant bit.
m †	Bit number of the highest bit set in the absolute value of register **A**. Bit 0 is the least significant bit.
n	Number of places shifted.
w	Number of words in the message. Part words are counted as full words. If the message is not word aligned the number of words is increased to include the part words at either end of the message.
p †	Number of words per row.
r †	Number of rows.
† does not apply to IMS T222 and IMS T225	

Table 1.4 Instruction set interpretation

The **DEF** column of the tables indicates the descheduling/error features of an instruction as described in table 1.5.

Ident	Feature	See section:
D	The instruction is a descheduling point	2
E	The instruction will affect the **Error** flag	3
F †	The instruction will affect the **FP_Error** flag	5
† applies to IMS T801 and IMS T805 only		

Table 1.5 Instruction features

2 Descheduling points

The instructions in table 2.1 are the only ones at which a process may be descheduled. They are also the ones at which the processor will halt if the **Analyse** pin is asserted (refer to **Analyse** section of specific device e.g. IMS T805 page 81).

input message	*output message*	*output byte*	*output word*
timer alt wait	*timer input*	*stop on error*	*alt wait*
jump	*loop end*	*end process*	*start process*

Table 2.1 Descheduling point instructions

3 Error instructions

The instructions in table 3.1 are the only ones which can affect the *Error* flag directly (refer to **Error, ErrorIn** section of specific device e.g. IMS T805 page 83). Note, however, that the floating point unit error flag *FP_Error* is set by certain floating point instructions (section 5), and that *Error* can be set from this flag by *fpcheckerror.*

add	*add constant*	*subtract*	
multiply	*fractional multiply* †	*divide*	*remainder*
long add	*long subtract*	*long divide*	
set error	*testerr*	*fpcheckerror* ‡	
check word	*check subscript from 0*	*check single*	*check count from 1*
† does not apply to IMS T222 and IMS T225			
‡ applies to IMS T801 and IMS T805 only			

Table 3.1 Error setting instructions

4 Debugging support

Table 7.14 (page 55) contains a number of instructions to facilitate the implementation of breakpoints. These instructions overload the operation of *j0*. Normally *j0* is a no-op which might cause descheduling. *Setj0break* enables the breakpointing facilities and causes *j0* to act as a breakpointing instruction. When breakpointing is enabled, *j0* swaps the current **Iptr** and **Wptr** with an **Iptr** and **Wptr** stored above MemStart. The *break* instruction does not cause descheduling, and preserves the state of the registers. It is possible to single step the processor at machine level using these instructions. Refer to *Support for debugging/breakpointing in transputers* (technical note 61) for more detailed information regarding debugger support.

5 Floating point errors for the IMS T801 and IMS T805 only

The FPU has its own error flag *FP_Error*. This reflects the state of evaluation within the FPU and is set in circumstances where invalid operations, division by zero or overflow exceptions to the ANSI-IEEE 754-1985 standard would be flagged. *FP_Error* is also set if an input to a floating point operation is infinite or is not a number (NaN). The *FP_Error* flag can be set, tested and cleared without affecting the main *Error* flag, but can also set *Error* when required. Depending on how a program is compiled, it is possible for both unchecked and fully checked floating point arithmetic to be performed.

The instructions in table 5.1 are the only ones which can affect the floating point error flag *FP_Error*. *Error* is set from this flag by *fpcheckerror* if *FP_Error* is set.

fpadd	*fpsub*	*fpmul*	*fpdiv*
fpldnladdsn	*fpldnladddb*	*fpldnlmulsn*	*fpldnlmuldb*
fpremfirst	*fpusqrtfirst*	*fpgt*	*fpeq*
fpuseterror	*fpuclearerror*	*fptesterror*	
fpuexpincby32	*fpuexpdecby32*	*fpumulby2*	*fpudivby2*
fpur32tor64	*fpur64tor32*	*fpucki32*	*fpucki64*
fprtoi32	*fpuabs*	*fpint*	

Table 5.1 Floating point error setting instructions

6 Block move

The block move instructions (Table 7.5) move any number of bytes from any byte boundary in memory, to any other byte boundary, using the smallest possible number of word read, and word or part-word writes.

A block move instruction can be interrupted by a high priority process. On interrupt, block move is completed to a word boundary, independent of start position. When restarting after interrupt, the last word written is written again. This appears as an unnecessary read and write in the simplest case of word aligned block moves, and may cause problems with FIFOs. This problem can be overcome by incrementing the saved destination (**BregIntSaveLoc**) and source pointer (**CregIntSaveLoc**) values by BytesPerWord during the high priority process.

7 General instructions

The following tables list the complete instruction set which is common to all variants of the transputer. Exceptions are noted at the bottom of each table by † or ‡.

Function Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
0	0X	j	3	jump	D
1	1X	ldlp	1	load local pointer	
2	2X	pfix	1	prefix	
3	3X	ldnl	2	load non–local	
4	4X	ldc	1	load constant	
5	5X	ldnlp	1	load non–local pointer	
6	6X	nfix	1	negative prefix	
7	7X	ldl	2	load local	
8	8X	adc	1	add constant	E
9	9X	call	7	call	
A	AX	cj	2	conditional jump (not taken)	
			4	conditional jump (taken)	
B	BX	ajw	1	adjust workspace	
C	CX	eqc	2	equals constant	
D	DX	stl	1	store local	
E	EX	stnl	2	store non-local	
F	FX	opr	–	operate	

Table 7.1 Function codes

Operation	Memory	Mnemonic	Processor Cycles		Name	DEF
Code	Code		16-bit devices	32-bit devices		
46	24F6	and	1	1	and	
4B	24FB	or	1	1	or	
33	23F3	xor	1	1	exclusive or	
32	23F2	not	1	1	bitwise not	
41	24F1	shl	**n**+2	**n**+2	shift left	
40	24F0	shr	**n**+2	**n**+2	shift right	
05	F5	add	1	1	add	E
0C	FC	sub	1	1	subtract	E
53	25F3	mul	23	38	multiply	E
72 †	27F2	fmul		35	fractional multiply (no rounding)	E
				40	fractional multiply (rounding)	E
2C	22FC	div	24	39	divide	E
1F	21FF	rem	21	37	remainder	E
09	F9	gt	2	2	greater than	
04	F4	diff	1	1	difference	
52	25F2	sum	1	1	sum	
08	F8	prod	**b**+4	**b**+4	product for positive register **A**	
08 ‡	F8	prod	**m**+5	**m**+5	product for negative register **A**	

† does not apply to IMS T222 and IMS T225

‡ does not apply to IMS T222

Table 7.2 Arithmetic/logical operation codes

Operation	Memory	Mnemonic	Processor Cycles		Name	DEF
Code	Code		16-bit devices	32-bit devices		
16	21F6	ladd	2	2	long add	E
38	23F8	lsub	2	2	long subtract	E
37	23F7	lsum	3	3	long sum	
4F	24FF	ldiff	3	3	long diff	
31	23F1	lmul	17	33	long multiply	
1A	21FA	ldiv	19	35	long divide	E
36	23F6	lshl	**n**+3	**n**+3	long shift left (**n**<32) † (**n**<16)	
			n–12	**n**–28	long shift left(**n**≥32) † (**n**≥16)	
35	23F5	lshr	**n**+3	**n**+3	long shift right (**n**<32) † (**n**<16)	
			n–12	**n**–28	long shift right (**n**≥32) † (**n**≥16)	
19	21F9	norm	**n**+5	**n**+5	normalise (**n**<32) † (**n**<16)	
			n–10	**n**–26	normalise (**n**≥32) † (**n**≥16)	
			3	3	normalise (**n**=64) † (**n**=32)	

† for IMS T222 and IMS T225

Table 7.3 Long arithmetic operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
00	F0	rev	1	reverse	
3A	23FA	xword	4	extend to word	
56	25F6	cword	5	check word	E
1D	21FD	xdble	2	extend to double	
4C	24FC	csngl	3	check single	E
42	24F2	mint	1	minimum integer	
5A †	25FA	dup	1	duplicate top of stack	
79 †	27F9	pop	1	pop processor stack	
† does not apply to IMS T222					

Table 7.4 General operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
5B †	25FB	move2dinit	8	initialise data for 2D block move	
5C †	25FC	move2dall	(2p+23)*r	2D block copy	
5D †	25FD	move2dnonzero	(2p+23)*r	2D block copy non-zero bytes	
5E †	25FE		(2p+23)*r	2D block copy zero bytes	
† does not apply to IMS T222 and IMS T225					

Table 7.5 2D block move operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
74 †	27F4	crcword	35	calculate crc on word	
75 †	27F5	crcbyte	11	calculate crc on byte	
76 †	27F6	bitcnt	**b**+2	count bits set in word	
77 †	27F7	bitrevword	36	reverse bits in word	
78 †	27F8	bitrevnbits	**n**+4	reverse bottom n bits in word	
† does not apply to IMS T222					

Table 7.6 CRC and bit operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles		Name	DEF
			16-bit devices	32-bit devices		
02	F2	bsub	1	1	byte subscript	
0A	FA	wsub	2	2	word subscript	
81 †	28F1	wsubdb	3	3	form double word subscript	
34	23F4	bcnt	2	2	byte count	
3F	23FF	wcnt	4	5	word count	
01	F1	lb	5	5	load byte	
3B	23FB	sb	4	4	store byte	
4A	24FA	move	2w+8	2w+8	move message	
† does not apply to IMS T222 and IMS T225						

Table 7.7 Indexing/array operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
22	22F2	ldtimer	2	load timer	
2B	22FB	tin	30	timer input (time future)	D
			4	timer input (time past)	D
4E	24FE	talt	4	timer alt start	
51	25F1	taltwt	15	timer alt wait (time past)	D
			48	timer alt wait (time future)	D
47	24F7	enbt	8	enable timer	
2E	22FE	dist	23	disable timer	

Table 7.8 Timer handling operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
07	F7	in	2w+19	input message	D
0B	FB	out	2w+19	output message	D
0F	FF	outword	23	output word	D
0E	FE	outbyte	23	output byte	D
43	24F3	alt	2	alt start	
44	24F4	altwt	5	alt wait (channel ready)	D
			17	alt wait (channel not ready)	D
45	24F5	altend	4	alt end	
49	24F9	enbs	3	enable skip	
30	23F0	diss	4	disable skip	
12	21F2	resetch	3	reset channel	
48	24F8	enbc	7	enable channel (ready)	
			5	enable channel (not ready)	
2F	22FF	disc	8	disable channel	

Table 7.9 Input/output operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
20	22F0	ret	5	return	
1B	21FB	ldpi	2	load pointer to instruction	
3C	23FC	gajw	2	general adjust workspace	
06	F6	gcall	4	general call	
21	22F1	lend	10	loop end (loop)	D
			5	loop end (exit)	D

Table 7.10 Control operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
0D	FD	startp	12	start process	
03	F3	endp	13	end process	D
39	23F9	runp	10	run process	
15	21F5	stopp	11	stop process	
1E	21FE	ldpri	1	load current priority	

Table 7.11 Scheduling operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
13	21F3	csub0	2	check subscript from 0	E
4D	24FD	ccnt1	3	check count from 1	E
29	22F9	testerr	2	test error false and clear (no error)	
			3	test error false and clear (error)	
10	21F0	seterr	1	set error	E
55	25F5	stoperr	2	stop on error (no error)	D
57	25F7	clrhalterr	1	clear halt–on–error	
58	25F8	sethalterr	1	set halt–on–error	
59	25F9	testhalterr	2	test halt–on–error	

Table 7.12 Error handling operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
2A	22FA	testpranal	2	test processor analyzing	
3E	23FE	saveh	4	save high priority queue registers	
3D	23FD	savel	4	save low priority queue registers	
18	21F8	sthf	1	store high priority front pointer	
50	25F0	sthb	1	store high priority back pointer	
1C	21FC	stlf	1	store low priority front pointer	
17	21F7	stlb	1	store low priority back pointer	
54	25F4	sttimer	1	store timer	
17C †	2127FC	lddevid	1	load device identity	
7E †	27FE	ldmemstartval	1	load value of memstart address	
† does not apply to IMS T222					

Table 7.13 Processor initialisation operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
0 †	00	jump 0	3	jump 0 (break not enabled)	D
			11	jump 0 (break enabled, high priority)	
			13	jump 0 (break enabled, low priority)	
B1 †	2BF1	break	9	break (high priority)	
			11	break (low priority)	
B2 †	2BF2	clrj0break	1	clear jump 0 break enable flag	
B3 †	2BF3	setj0break	1	set jump 0 break enable flag	
B4 †	2BF4	testj0break	2	test jump 0 break enable flag set	
7A †	27FA	timerdisableh	1	disable high priority timer interrupt	
7B †	27FB	timerdisablel	1	disable low priority timer interrupt	
7C †	27FC	timerenableh	6	enable high priority timer interrupt	
7D †	27FD	timerenablel	6	enable low priority timer interrupt	
† does not apply to IMS T222					

Table 7.14 Debugger support codes

8 Floating point instructions

8.1 Floating point instructions for IMS T801 and IMS T805 only

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
8E	28FE	fpldnlsn	2	fp load non-local single	
8A	28FA	fpldnldb	3	fp load non-local double	
86	28F6	fpldnlsni	4	fp load non-local indexed single	
82	28F2	fpldnldbi	6	fp load non-local indexed double	
9F	29FF	fpldzerosn	2	load zero single	
A0	2AF0	fpldzerodb	2	load zero double	
AA	2AFA	fpldnladdsn	8/11	fp load non local & add single	F
A6	2AF6	fpldnladddb	9/12	fp load non local & add double	F
AC	2AFC	fpldnlmulsn	13/20	fp load non local & multiply single	F
A8	2AF8	fpldnlmuldb	21/30	fp load non local & multiply double	F
88	28F8	fpstnlsn	2	fp store non-local single	
84	28F4	fpstnldb	3	fp store non-local double	
9E	29FE	fpstnli32	4	store non-local int32	
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 8.1 Floating point load/store operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
AB	2AFB	fpentry	1	floating point unit entry	
A4	2AF4	fprev	1	fp reverse	
A3	2AF3	fpdup	1	fp duplicate	

Table 8.2 Floating point general operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
22	s	fpurn	1	set rounding mode to round nearest	
06	s	fpurz	1	set rounding mode to round zero	
04	s	fpurp	1	set rounding mode to round positive	
05	s	fpurm	1	set rounding mode to round minus	

Table 8.3 Floating point rounding operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
83	28F3	fpchkerror	1	check fp error	E
9C	29FC	fptesterror	2	test fp error false and clear	F
23	s	fpuseterror	1	set fp error	F
9C	s	fpuclearerror	1	clear fp error	F

Table 8.4 Floating point error operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
94	29F4	fpgt	4/6	fp greater than	F
95	29F5	fpeq	3/5	fp equality	F
92	29F2	fpordered	3/4	fp orderability	
91	29F1	fpnan	2/3	fp NaN	
93	29F3	fpnotfinite	2/2	fp not finite	
0E	s	fpuchki32	3/4	check in range of type int32	F
0F	s	fpuchki64	3/4	check in range of type int64	F
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 8.5 Floating point comparison operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
07	s	fpur32tor64	3/4	real32 to real64	F
08	s	fpur64tor32	6/9	real64 to real32	F
9D	29FD	fprtoi32	7/9	real to int32	F
96	29F6	fpi32tor32	8/10	int32 to real32	
98	29F8	fpi32tor64	8/10	int32 to real64	
9A	29FA	fpb32tor64	8/8	bit32 to real64	
0D	s	fpunoround	2/2	real64 to real32, no round	
A1	2AF1	fpint	5/6	round to floating integer	F
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 8.6 Floating point conversion operation codes

Operation	Memory		Processor Cycles			
Code	Code	Mnemonic	Single	Double	Name	DEF
87	28F7	fpadd	6/9	6/9	fp add	F
89	28F9	fpsub	6/9	6/9	fp subtract	F
8B	28FB	fpmul	11/18	18/27	fp multiply	F
8C	28FC	fpdiv	16/28	31/43	fp divide	F
0B	**s**	fpuabs	2/2	2/2	fp absolute	F
8F	28FF	fpremfirst	36/46	36/46	fp remainder first step	F
90	29F0	fpremstep	32/36	32/36	fp remainder iteration	
01	**s**	fpusqrtfirst	27/29	27/29	fp square root first step	F
02	**s**	fpusqrtstep	42/42	42/42	fp square root step	
03	**s**	fpusqrtlast	8/9	8/9	fp square root end	
0A	**s**	fpuexpinc32	6/9	6/9	multiply by 2^{32}	F
09	**s**	fpuexpdec32	6/9	6/9	divide by 2^{32}	F
12	**s**	fpumulby2	6/9	6/9	multiply by 2.0	F
11	**s**	fpudivby2	6/9	6/9	divide by 2.0	F
Processor cycles are shown as **Typical/Maximum** cycles.						

Table 8.7 Floating point arithmetic operation codes

8.2 Floating point instructions for IMS T400, IMS T425 and IMS T426 only

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
73	27F3	cflerr	3	check floating point error	E
9C	29FC	fptesterr	1	load value true (FPU not present)	
63	26F3	unpacksn	15	unpack single length fp number	
6D	26FD	roundsn	12/15	round single length fp number	
6C	26FC	postnormsn	5/30	post–normalise correction of single length fp number	
71	27F1	ldinf	1	load single length infinity	
Processor cycles are shown as **Typical/Maximum** cycles.					

Table 8.8 Floating point support operation codes

Transputer performance

1 Introduction

The performance of the transputer is measured in terms of the number of bytes required for the program, and the number of (internal) processor cycles required to execute the program. The figures here relate to occam programs. For the same function, other languages should achieve approximately the same performance as occam.

With transputers incorporating an FPU, this type of performance calculation is straight forward when considering only integer data types. However, when floating point calculations using the REAL32 and REAL64 data types are present in the program, complications arise due to the concurrency inherent in the transputer's design whereby integer calculations can be overlapped with floating point calculations. A more comprehensive guide to the impact of this concurrency on transputer performance can be found in the *Transputer Instruction Set – A Compiler Writer's Guide*.

2 Performance overview

These figures are averages obtained from detailed simulation, and should be used only as an initial guide; they assume operands are of type INT. The abbreviations in table 2.1 are used to represent the quantities indicated. In the replicator section of the table, figures in braces {} are not necessary if the number of replications is a compile time constant. To estimate performance, add together the time for the variable references and the time for the operation.

np	number of component processes
ne	number of processes earlier in queue
r	1 if INT parameter or array parameter, 0 if not
ts	number of table entries (table size)
w	width of constant in nibbles
p	number of places to shift
Eg	expression used in a guard
Et	timer expression used in a guard
Tb	most significant bit set of multiplier ((–1) if the multiplier is 0)
Tbp	most significant bit set in a positive multiplier when counting from zero ((–1) if the multiplier is 0)
Tbc	most significant bit set in the two's complement of a negative multiplier
nsp	number of scalar parameters in a procedure
nap	number of array parameters in a procedure

Table 2.1 Key to performance table

	Size (bytes)	Time (cycles)
Names		
variables		
in expression	1.1 + **r**	2.1 + 2 (**r**)
assigned to or input to	1.1 + **r**	1.1 + (**r**)
in PROC or FUNCTION call,		
corresponding to an INT parameter	1.1 + **r**	1.1 + (**r**)
channels	1.1	2.1
Array Variables (for single dimension arrays)		
constant subscript	0	0
variable subscript	5.3	7.3
expression subscript	5.3	7.3
Declarations		
CHAN OF *protocol*	3.1	3.1
[size] CHAN OF *protocol*	9.4	2.2 + 20.2 * size
PROC	body + 2	0
Primitives		
assignment	0	0
input	4	26.5
output	1	26
STOP	2	25
SKIP	0	0
Arithmetic operators		
+ –	1	1
*	2	39
/	2	40
REM	2	38
>> <<	2	3 + **p**
Modulo Arithmetic operators		
PLUS	2	2
MINUS	1	1
TIMES (fast multiply, positive operand)	1	4 + **Tbp**
TIMES (fast multiply, negative operand) †	1	5 + **Tbc**
Boolean operators		
OR	4	8
AND NOT	1	2
Comparison operators		
= constant	0	1
= variable	2	3
<> constant	1	3

† not IMS T222

table continued on next page

table continued from previous page

	Size (bytes)	Time (cycles)
Comparison operators (cont'd)		
<> variable	3	5
> <	1	2
> = < =	2	4
Bit operators		
/\ \/ >< ~	2	2
Expressions		
constant in expression	**w**	**w**
check if error	4	6
Timers		
timer input	2	3
timer `AFTER`		
if past time	2	4
with empty timer queue	2	31
non–empty timer queue	2	38 + **ne** * 9
`ALT` (timer)		
with empty timer queue	6	52
non–empty timer queue	6	59 + **ne** * 9
timer alt guard	8 + 2 **Eg** + 2 **Et**	34 + 2 **Eg** + 2 **Et**
Constructs		
`SEQ`	0	0
`IF`	1.3	1.4
if guard	3	4.3
`ALT` (non timer)	6	26
alt channel guard	10.2 + 2 **Eg**	20 + 2 **Eg**
skip alt guard	8 + 2 **Eg**	10 + 2 **Eg**
`PAR`	11.5 + (**np**–1) * 7.5	19.5 + (**np**–1) * 30.5
`WHILE`	4	12
Procedure or function call		
	3.5 + (**nsp**–2) * 1.1 + **nap** * 2.3	16.5 + (**nsp**–2) * 1.1 + **nap** * 2.3
Replicators		
replicated `SEQ`	7.3 {+5.1}	(–3.8)+15.1*count {+7.1}
replicated `IF`	12.3 {+5.1}	(–2.6)+19.4*count {+7.1}
replicated `ALT`	24.8 {+10.2}	25.4+33.4*count {+14.2}
replicated timer `ALT`	24.8 {+10.2}	62.4+33.4*count {+14.2}
replicated `PAR`	39.1 {+5.1}	(–6.4)+70.9*count {+7.1}

Table 2.2 Performance

3 Fast multiply, TIMES

The transputer has a fast integer multiplication instruction *product*. For a positive multiplier its execution time is 4+**Tbp** cycles, and for a negative multiplier 5+**Tbc** cycles (table 2.1). The time taken for a multiplication by zero is 3 cycles.

Implementations of high level languages on the transputer may take advantage of this instruction. For example, the occam modulo arithmetic operator TIMES is implemented by the instruction and the right-hand operand is treated as the multiplier. The fast multiplication instruction is also used in high level language implementations for the multiplication implicit in multi-dimensional array access.

4 Arithmetic

A set of functions are provided within the development system to support the efficient implementation of multiple length integer arithmetic. In the transputer, floating point arithmetic is taken care of by the FPU. In table 4.1 **n** gives the number of places shifted and all arguments and results are assumed to be local. Full details of these functions are provided in the occam reference manual, supplied as part of the development system and available as a separate publication.

When calculating the execution time of the predefined maths functions, no time needs to be added for calling overhead. These functions are compiled directly into special purpose instructions which are designed to support the efficient implementation of multiple length integer arithmetic and floating point arithmetic.

Function		Cycles	+ cycles for parameter access †
LONGADD		2	7
LONGSUM		3	8
LONGSUB		2	7
LONGDIFF		3	8
LONGPROD		34	8
LONGDIV		36	8
SHIFTRIGHT	($\mathbf{n} < 32$)	$4 + \mathbf{n}$	8
	($\mathbf{n} \geq 32$)	$\mathbf{n} - 27$	
SHIFTLEFT	($\mathbf{n} < 32$)	$4 + \mathbf{n}$	8
	($\mathbf{n} \geq 32$)	$\mathbf{n} - 27$	
NORMALISE	($\mathbf{n} < 32$)	$\mathbf{n} + 6$	7
	($\mathbf{n} \geq 32$)	$\mathbf{n} - 25$	
	($\mathbf{n} = 64$)	4	
ASHIFTRIGHT		SHIFTRIGHT + 2	5
ASHIFTLEFT		SHIFTLEFT + 4	5
ROTATERIGHT		SHIFTRIGHT	7
ROTATELEFT		SHIFTLEFT	7
FRACMUL		LONGPROD+4	5

† Assuming local variables.

Table 4.1 32-bit arithmetic performance

Function		Cycles	+ cycles for parameter access †
LONGADD		2	7
LONGSUM		3	8
LONGSUB		2	7
LONGDIFF		3	8
LONGPROD		18	8
LONGDIV		20	8
SHIFTRIGHT	($n < 16$)	$4 + n$	8
	($n \geq 16$)	$n - 11$	8
SHIFTLEFT	($n < 16$)	$4 + n$	8
	($n \geq 16$)	$n - 11$	8
NORMALISE	($n < 16$)	$n + 6$	7
	($n \geq 16$)	$n - 9$	7
	($n = 32$)	4	7
ASHIFTRIGHT		SHIFTRIGHT + 2	5
ASHIFTLEFT		SHIFTLEFT + 4	5
ROTATERIGHT		SHIFTRIGHT	7
ROTATELEFT		SHIFTLEFT	7
† Assuming local variables.			

Table 4.2 16-bit arithmetic performance

5 Floating point operations

5.1 Floating point operations for IMS T801 and IMS T805 only

All references to REAL32 or REAL64 operands within programs compiled for the transputer normally produce the following performance figures.

	Size (bytes)	REAL32 Time (cycles)	REAL64 Time (cycles)
Names			
variables			
in expression	3.1	3	5
assigned to or input to	3.1	3	5
in PROC or FUNCTION call, corresponding to a REAL parameter	1.1+r	1.1+r	1.1+r
Arithmetic operators			
+ -	2	7	7
*	2	11	20
/	2	17	32
REM	11	19	34
Comparison operators			
=	2	4	4
<>	3	6	6
> <	2	5	5
>= <=	3	7	7
Conversions			
REAL32 to –	2		3
REAL64 to –	2	6	
To INT32 from –	5	9	9
To INT64 from –	18	32	32
INT32 to –	3	7	7
INT64 to –	14	24	22

Table 5.1 IMS T801 and IMS T805 floating point performance

5.1.1 Floating point functions

These functions are provided by the development system. They are compiled directly into special purpose instructions designed to support the efficient implementation of some of the common mathematical functions of other languages. The functions provide ABS and SQRT for both REAL32 and REAL64 operand types.

		+ cycles for parameter access †	
Function	**Cycles**	REAL32	REAL64
ABS	2	8	
SQRT	118	8	
DABS	2		12
DSQRT	244		12
† Assuming local variables.			

Table 5.2 Floating point arithmetic performance

5.2 Floating point operations for IMS T222 and IMS T225

Floating point operations for the IMS T222 and IMS T225 are provided by a run-time package. This requires approximately 2000 bytes of memory for the double length arithmetic operations, and 2500 bytes for the quadruple length arithmetic operations. Table 5.3 summarizes the estimated performance of the package.

		Processor cycles	
		Typical	Worst
REAL32	+ -	530	705
	*	650	705
	/	1000	1410
	< > = >= <= <>	60	60
REAL64	+ -	875	1190
	*	1490	1950
	/	2355	3255
	< > = >= <= <>	60	60

Table 5.3 IMS T222 and IMS T225 floating point performance

5.3 Floating point operations for IMS T400, IMS T425 and IMS T426

Floating point operations for the IMS T400, IMS T425 and IMS T426 are provided by a run-time package. This requires approximately 400 bytes of memory for the single length arithmetic operations, and 2500 bytes for the double length arithmetic operations. Table 5.4 summarizes the estimated performance of the package.

		Processor cycles	
		Typical	Worst
REAL32	+ -	230	300
	*	200	240
	/	245	280
	< > = >= <= <>	60	60
REAL64	+ -	565	700
	*	760	940
	/	1115	1420
	< > = >= <= <>	60	60

Table 5.4 IMS T400, IMS T425 and IMS T426 floating point performance

5.4 Special purpose functions and procedures

The functions and procedures given in tables 5.6 and 5.7 are provided by the development system to give access to the special instructions available on the transputer. Table 5.5 shows the key to the table.

Tb	most significant bit set in the word counting from zero
n	number of words per row (consecutive memory locations)
r	number of rows in the two dimensional move
nr	number of bits to reverse

Table 5.5 Key to special performance table

Function	**Cycles**	**+ cycles for parameter access †**
BITCOUNT ‡	2 + **Tb**	2
CRCBYTE ‡	11	8
CRCWORD ‡	35	8
BITREVNBIT ‡	5 + **nr**	4
BITREVWORD ‡	36	2

† Assuming local variables.

‡ Not IMS T222

Table 5.6 Special purpose functions performance

Function	**Cycles**	**+ cycles for parameter access †**
MOVE2D ‡	8+(2**n**+23)***r**	8
DRAW2D ‡	8+(2**n**+23)***r**	8
CLIP2D ‡	8+(2**n**+23)***r**	8

† Assuming local variables.

‡ Not IMS T222, IMS T225

Table 5.7 Special purpose procedures performance

6 Effect of external memory

Extra processor cycles may be needed when program and/or data are held in external memory, depending both on the operation being performed, and on the speed of the external memory. After a processor cycle which initiates a write to memory, the processor continues execution at full speed until at least the next memory access.

Whilst a reasonable estimate may be made of the effect of external memory, the actual performance will depend upon the exact nature of the given sequence of operations.

External memory is characterized by the number of extra processor cycles per external memory cycle, denoted as **e**.

> For the IMS T805, IMS T400, IMS T425, IMS T426 with the fastest external memory the value of **e** is 2; a typical value for a large external memory is 5.
> The value of **e** for the IMS T801, IMS T222 and IMS T225 with no wait states is 1.

If a program is stored in external memory, and **e** has the value 2 or 3, then no extra cycles need be estimated for linear code sequences. For larger values of **e**, the number of extra cycles required for linear code sequences may be estimated at:

(**e**–3)/4 for 32-bit transputers.
(2**e**-1)/4 per byte of program for 16-bit transputers.

A transfer of control may be estimated as requiring **e**+3 cycles.

These estimates may be refined for various constructs. In tables 6.1 and 6.2 **n** denotes the number of components in a construct. In the case of `IF`, the **n**'th conditional is the first to evaluate to `TRUE`, and the costs include the costs of the conditionals tested. The number of bytes in an array assignment or communication is denoted by **b**.

	Program off chip	Data off chip
Boolean expressions	**e** – 2	0
`IF`	3**en** – 8	**en**
Replicated `IF`	(6**e** – 4) **n** + 7	(5**e** – 2) **n** + 8
Replicated `SEQ`	(3**e** – 3) **n** + 2	(4**e** – 2) **n**
`PAR`	(3**e** – 1) **n** + 8	3**en** + 4
Replicated `PAR`	(10**e** – 8) **n** + 8	16**en** – 12
`ALT`	(2**e** – 4) **n** +6**e**	(2**e** – 2) **n** +10**e** – 8
Array assignment and communication in one transputer	0	max (2**e**, **e** (**b**/2))

Table 6.1 External memory performance for 32-bit transputers

	Program off chip	Data off chip
Boolean expressions	**e** – 1	0
`IF`	3**en** – 1	**en**
Replicated `IF`	6**en** + 9**e** – 12	(5**e** – 2) **n** + 6
Replicated `SEQ`	(4**e** – 3) **n** + 3**e**	(4**e** – 2) **n** + 3 – **e**
`PAR`	4**en**	3**en**
Replicated `PAR`	(17**e** – 12) **n** + 9	16**en**
`ALT`	(4**e** – 1) **n** +9**e** – 4	(4**e** – 1) **n** +9**e** – 3
Array assignment and communication in one transputer	0	max (2**e**, **eb**)

Table 6.2 External memory performance for 16-bit transputers

The following simulation results illustrate the effect of storing program and/or data in external memory. The results are normalized to 1 for both program and data on chip. The first program (Sieve of Erastosthenes) is an extreme case as it is dominated by small, data access intensive loops; it contains no concurrency, communication, or even multiplication or division. The second program is the pipeline algorithm for Newton Raphson square rootcomputation.

	Program	e=2	e=3	e=4	e=5	On chip
Program off chip	1	1.3	1.5	1.7	1.9	1
	2	1.1	1.2	1.2	1.3	1
Data off chip	1	1.5	1.8	2.1	2.3	1
	2	1.2	1.4	1.6	1.7	1
Program and data off chip	1	1.8	2.2	2.7	3.2	1
	2	1.3	1.6	1.8	2.0	1

Table 6.3 External memory performance for 32-bit transputers

	Program	e=2	e=3	e=4	e=5	On chip
Program off chip	1	1.2	1.4	1.8	2.1	1
	2	1.1	1.2	1.4	1.6	1
Data off chip	1	1.2	1.5	1.8	2.1	1
	2	1.2	1.3	1.4	1.6	1
Program and data off chip	1	1.4	1.9	2.5	3.0	1
	2	1.2	1.5	1.8	2.1	1

Table 6.4 External memory performance for 16-bit transputers

7 Interrupt latency

If the process is a high priority one and no other high priority process is running, the latency is as described in table 7.1. The timings given are in full processor cycles **TPCLPCL**; the number of **Tm** states is also given where relevant. Maximum latency assumes all memory accesses are internal ones.

	Typical		**Maximum**	
	TPCLPCL	**Tm**	**TPCLPCL**	**Tm**
IMS T222, IMS T225	19		53	
IMS T400, IMS T425, IMS T426	19	38	58	116
IMS T801, IMS T805				
with FPU in use	19	38	78	156
with FPU not in use	19	38	58	116

Table 7.1 Interrupt latency

IMS T805 transputer

inmos®

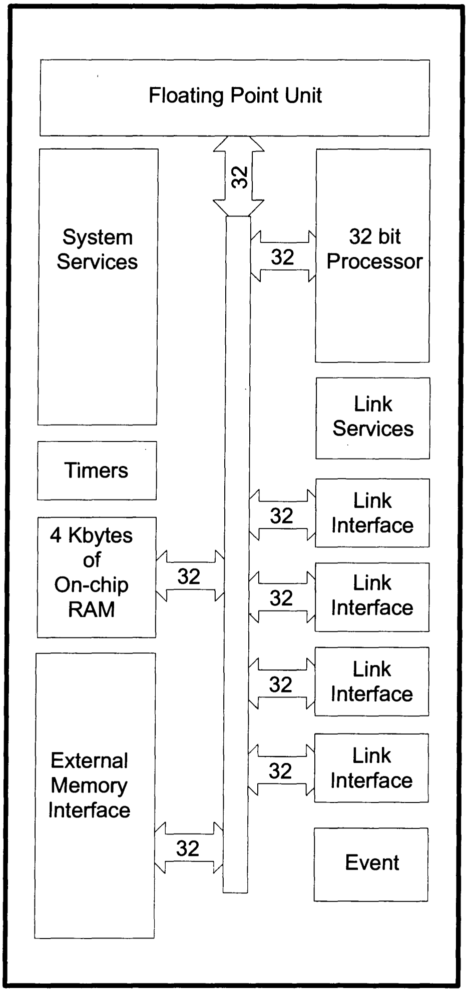
Engineering Data

FEATURES

32 bit architecture
33 ns internal cycle time
30 MIPS (peak) instruction rate
4.3 Mflops (peak) instruction rate
Pin compatible with IMS T800, IMS T425, IMS T400 and IMS T414
Debugging support
64 bit on-chip floating point unit which conforms to IEEE 754
4 Kbytes on-chip static RAM
120 Mbytes/sec sustained data rate to internal memory
4 Gbytes directly addressable external memory
40 Mbytes/sec sustained data rate to external memory
630 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
Bi-directional data rate of 2.4 Mbytes/sec per link
High performance graphics support with block move instructions
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ±5% power supply
Packaging 84 pin PGA / 84 pin PLCC / 100 pin CQFP
MIL-STD-883 processing will be available

APPLICATIONS

Scientific and mathematical applications
High speed multi processor systems
High performance graphics processing
Supercomputers
Workstations and workstation clusters
Digital signal processing
Accelerator processors
Distributed databases
System simulation
Telecommunications
Robotics
Fault tolerant systems
Image processing
Pattern recognition
Artificial intelligence



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992

1 Introduction

The IMS T805 transputer is a 32 bit CMOS microcomputer with a 64 bit floating point unit and graphics support. It has 4 Kbytes on-chip RAM for high speed processing, a configurable memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages and provides direct support for the occam model of concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond.

For convenience of description, the IMS T805 operation is split into the basic blocks shown in figure 1.1.

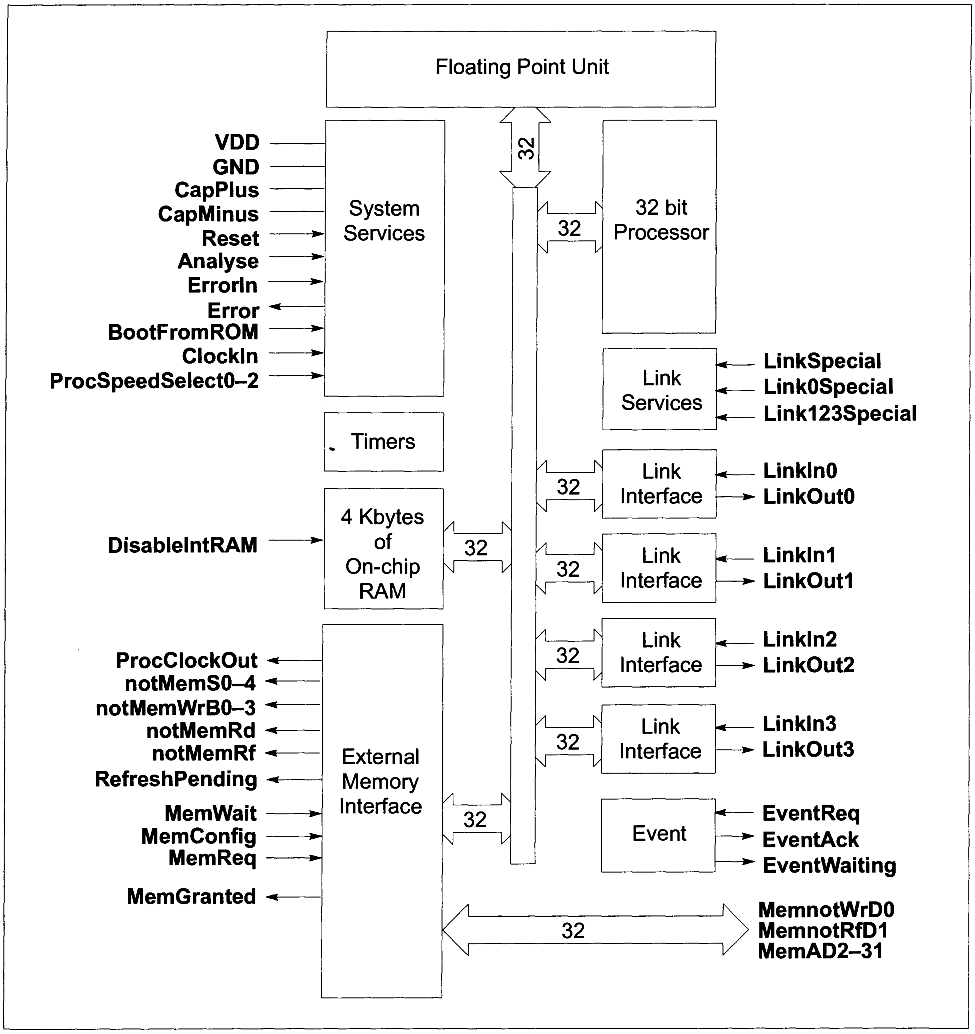


Figure 1.1 IMS T805 block diagram

The processor speed of a device can be pin-selected in stages from 20 MHz up to the maximum allowed for the part. A device running at 30 MHz achieves an instruction throughput of 30 MIPS peak and 15 MIPS sustained.

The IMS T805 provides high performance arithmetic and floating point operations. The 64 bit floating point unit provides single and double length operation to the ANSI-IEEE 754-1985 standard for floating point arithmetic. It is able to perform floating point operations concurrently with the processor, sustaining a rate of 2.2 Mflops at a processor speed of 20 MHz and 3.3 Mflops at 30 MHz.

High performance graphics support is provided by microcoded block move instructions which operate at the speed of memory. The two-dimensional block move instructions provide for contiguous block moves as well as block copying of either non-zero bytes of data only or zero bytes only. Block move instructions can be used to provide graphics operations such as text manipulation, windowing, panning, scrolling and screen updating.

Cyclic redundancy checking (CRC) instructions are available for use on arbitrary length serial data streams, to provide error detection where data integrity is critical. Another feature of the IMS T805, useful for pattern recognition, is the facility to count bits set in a word.

The IMS T805 can directly access a linear address space of 4 Gbytes. The 32 bit wide memory interface uses multiplexed data and address lines and provides a data rate of up to 4 bytes every 100 nanoseconds (40 Mbytes/sec) for a 30 MHz device. A configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems.

System Services include processor reset and bootstrap control, together with facilities for error analysis. Error signals may be daisy-chained in multi-transputer systems. The standard INMOS communication links allow networks of transputer family products to be constructed by direct point to point connections with no external logic. The IMS T805 links support the standard operating speed of 10 Mbits/sec, but also operate at 5 or 20 Mbits/sec. Each link can transfer data bi-directionally at up to 2.35 Mbytes/sec.

The IMS T805 is pin compatible with the IMS T800, as the extra inputs used are all held to ground on the IMS T800. The IMS T805-20 can thus be plugged directly into a circuit designed for a 20 MHz version of the IMS T800, and the IMS T805-25 can be used in place of a 25 MHz IMS T800.

The transputer is designed to implement the occam language, detailed in the occam Reference Manual, but also efficiently supports other languages such as C, Pascal and Fortran. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*. The instruction set of the IMS T805 is a superset of the IMS T800. Additional instructions are listed in Chapter 4.

This data sheet supplies hardware implementation and characterization details for the IMS T805. It is intended to be read in conjunction with the Transputer Architecture chapter, which details the architecture of the transputer and gives an overview of occam.

The IMS T805 instruction set contains a number of instructions to facilitate the implementation of breakpoints. For further information concerning breakpointing, refer to *Support for debugging/breakpointing in transputers* (technical note 61).

Figure 1.2 shows the internal datapaths for the IMS T805.

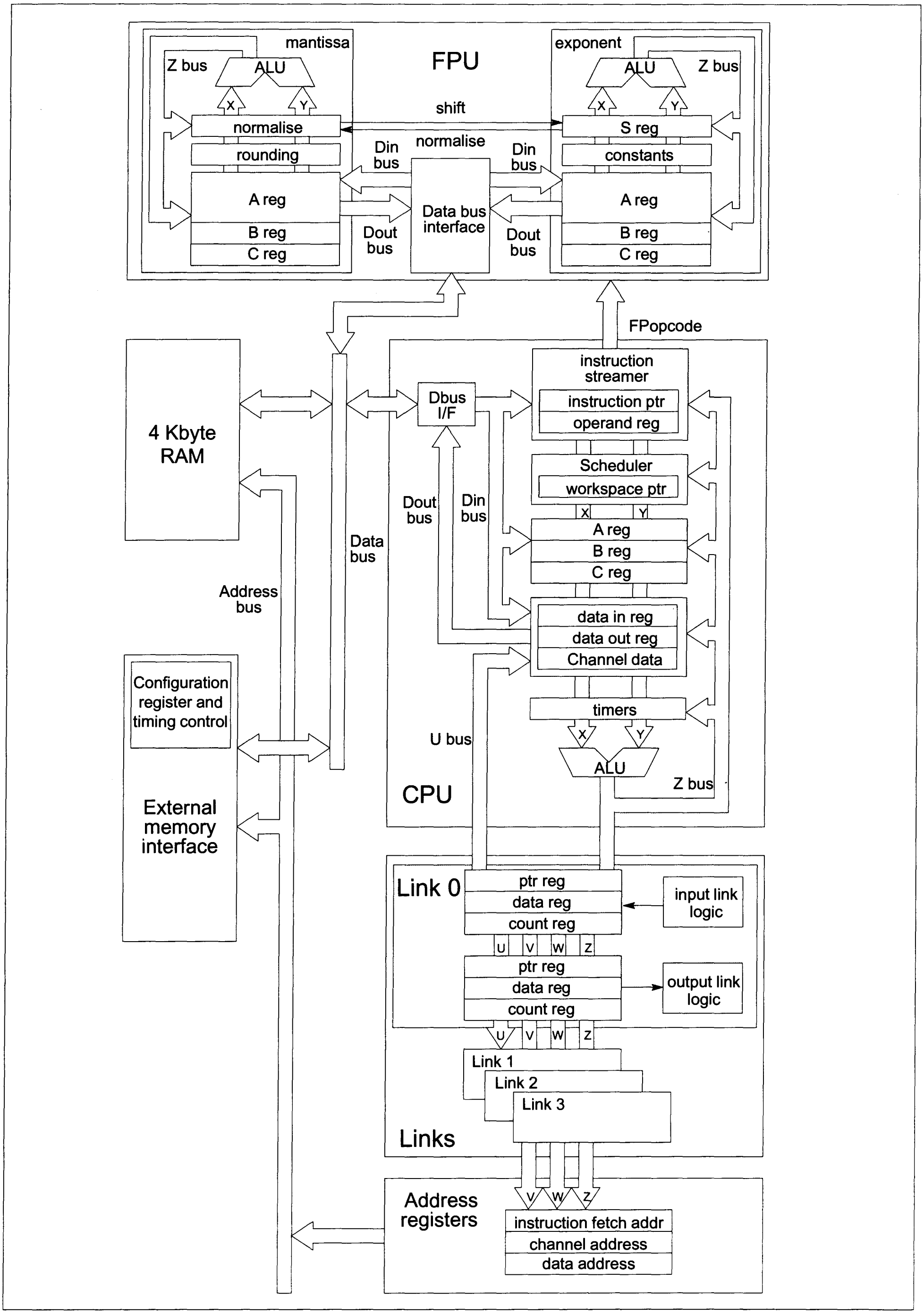


Figure 1.2 IMS T805 internal datapaths

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high.
Pinout details for various packages are given on page 122.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
ProcSpeedSelect0-2	in	Processor speed selectors
Reset	in	System reset
Error	out	Error indicator
ErrorIn	in	Error daisychain input
Analyse	in	Error analysis
BootFromROM	in	Boot from external ROM or from link
DisableIntRAM	in	Disable internal RAM

Table 2.1 IMS T805 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemnotWrD0	in/out	Multiplexed data bit 0 and write cycle warning
MemnotRfD1	in/out	Multiplexed data bit 1 and refresh warning
MemAD2-31	in/out	Multiplexed data and address bus
notMemRd	out	Read strobe
notMemWrB0-3	out	Four byte-addressing write strobes
notMemS0-4	out	Five general purpose strobes
notMemRf	out	Dynamic memory refresh indicator
RefreshPending	out	Dynamic refresh is pending
MemWait	in	Memory cycle extender
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted
MemConfig	in	Memory configuration data input

Table 2.2 IMS T805 external memory interface

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge
EventWaiting	out	Event input requested by software

Table 2.3 IMS T805 event

Pin	In/Out	Function
LinkIn0-3	in	Four serial data input channels
LinkOut0-3	out	Four serial data output channels
LinkSpecial	in	Select non-standard speed as 5 or 20 Mbits/sec
Link0Special	in	Select special speed for Link 0
Link123Special	in	Select special speed for Links 1,2,3

Table 2.4 IMS T805 link

3 Floating point unit

The 64 bit FPU provides single and double length arithmetic to floating point standard ANSI-IEEE 754-1985. It is able to perform floating point arithmetic concurrently with the central processor unit (CPU), sustaining 3.3 Mflops on a 30 MHz device. All data communication between memory and the FPU occurs under control of the CPU.

The FPU consists of a microcoded computing engine with a three deep floating point evaluation stack for manipulation of floating point numbers. These stack registers are **FA**, **FB** and **FC**, each of which can hold either 32 bit or 64 bit data; an associated flag, set when a floating point value is loaded, indicates which. The stack behaves in a similar manner to the CPU stack (page 27).

As with the CPU stack, the FPU stack is not saved when rescheduling (page 30) occurs. The FPU can be used in both low and high priority processes. When a high priority process interrupts a low priority one the FPU state is saved inside the FPU. The CPU will service the interrupt immediately on completing its current operation. The high priority process will not start, however, before the FPU has completed its current operation.

Points in an instruction stream where data need to be transferred to or from the FPU are called *synchronisation points*. At a synchronisation point the first processing unit to become ready will wait until the other is ready. The data transfer will then occur and both processors will proceed concurrently again. In order to make full use of concurrency, floating point data source and destination addresses can be calculated by the CPU whilst the FPU is performing operations on a previous set of data. Device performance is thus optimised by minimising the CPU and FPU idle times.

The FPU has been designed to operate on both single length (32 bit) and double length (64 bit) floating point numbers, and returns results which fully conform to the ANSI-IEEE 754-1985 floating point arithmetic standard. Denormalised numbers are fully supported in the hardware. All rounding modes defined by the standard are implemented, with the default being rounded to the nearest.

The basic addition, subtraction, multiplication and division operations are performed by single instructions. However, certain less frequently used floating point instructions are selected by a value in register **A** (when allocating registers, this should be taken into account). A *load constant* instruction *ldc* is used to load register **A**; the *floating point entry* instruction *fpentry* then uses this value to select the floating point operation. This pair of instructions is termed a *selector sequence*.

Names of operations which use *fpentry* begin with *fpu*. A typical usage, returning the absolute value of a floating point number, would be

fpuabs;   *ldc*   *fpentry;*

Since the indirection code for *fpuabs* is **0B**, it would be encoded as

Mnemonic		**Function code**	**Memory code**
ldc	*fpuabs*	#4	#4B
fpentry	(op. code #AB)		#2AFB
is coded as			
pfix	#A	#2	#2A
opr	#B	#F	#FB

Table 3.1 *fpentry* coding

The *remainder* and *square root* instructions take considerably longer than other instructions to complete. In order to minimise the interrupt latency period of the transputer they are split up to form instruction sequences. As an example, the instruction sequence for a single length square root is

fpusqrtfirst;   *fpusqrtstep;*   *fpusqrtstep;*   *fpusqrtlast;*

The FPU has its own error flag *FP_Error*. This reflects the state of evaluation within the FPU and is set in circumstances where invalid operations, division by zero or overflow exceptions to the ANSI-IEEE 754-1985 standard would be flagged (page 49). *FP_Error* is also set if an input to a floating point operation is infinite or is not a number (NaN). The *FP_Error* flag can be set, tested and cleared without affecting the main *Error* flag, but can also set *Error* when required (page 48). Depending on how a program is compiled, it is possible for both unchecked and fully checked floating point arithmetic to be performed.

Further details on the operation of the FPU can be found in the '*Transputer Instruction Set – A Compiler Writer's Guide*'.

	T805-20		**T805-25**		**T805-30**	
Operation	**Single length**	**Double length**	**Single length**	**Double length**	**Single length**	**Double length**
add	350 ns	350 ns	280 ns	280 ns	233 ns	233 ns
subtract	350 ns	350 ns	280 ns	280 ns	233 ns	233 ns
multiply	550 ns	1000 ns	440 ns	800 ns	367 ns	667 ns
divide	850 ns	1600 ns	680 ns	1280 ns	567 ns	1067 ns
Timing is for operations where both operands are normalised fp numbers.						

Table 3.2 Typical floating point operation times for IMS T805

4 System services

System services include all the necessary logic to initialise and sustain operation of the device. They also include error handling and analysis facilities.

4.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimise inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

4.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 10 MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

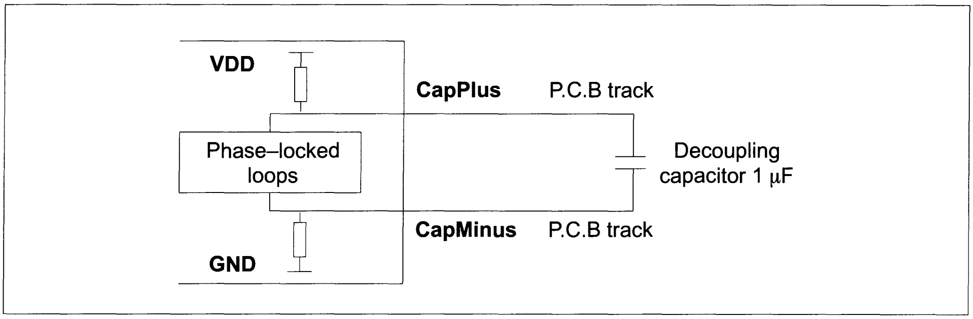


Figure 4.1 Recommended PLL decoupling

4.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	T805-20, -25, -30			Units	Notes
		Min	Nom	Max		
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1, 2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1, 3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1, 4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1. Guaranteed, but not tested.
2. Measured between corresponding points on consecutive falling edges.
3. Variation of individual falling edges from their nominal times.
4. This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.
5. Clock transitions must be monotonic within the range **VIH** to **VIL** (table 9.3).

Table 4.1 Input clock

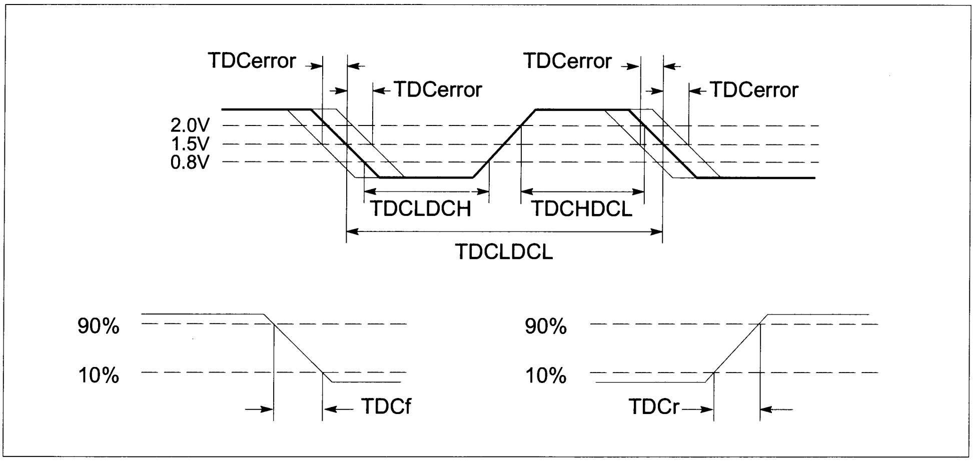


Figure 4.2 ClockIn timing

4.4 ProcSpeedSelect0–2

Processor speed of the IMS T805 is variable in discrete steps. The desired speed can be selected, up to the maximum rated for a particular component, by the three speed select lines **ProcSpeedSelect0-2**. The pins are tied high or low, according to table 4.2, for the various speeds. The pins are arranged so that the IMS T805 can be plugged directly into a board designed for a IMS T425.

Only six of the possible speed select combinations are currently used; the other two are not valid speed selectors. The frequency of **ClockIn** for the speeds given in the table is 5 MHz.

ProcSpeed-Select2	ProcSpeed-Select1	ProcSpeed-Select0	Processor Clock Speed MHz	Processor Cycle Time ns	Notes
0	0	0	20.0	50.0	
0	0	1	22.5	44.4	Not supported
0	1	0	25.0	40.0	
0	1	1	30.0	33.3	
1	0	0	35.0	28.6	Not supported
1	0	1			Invalid
1	1	0	17.5	57.1	Not supported
1	1	1			Invalid

Table 4.2 Processor speed selection

4.5 Bootstrap

The transputer can be bootstrapped either from a link or from external ROM. To facilitate debugging, **BootFromROM** may be dynamically changed but must obey the specified timing restrictions. It is sampled once only by the transputer, before the first instruction is executed after **Reset** is taken low.

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFFFFFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the *W* register points to *MemStart* (page 84).

If **BootFromROM** is connected low (e.g. to **GND**) the transputer will wait for the first bootstrap message to arrive on any one of its links. The transputer is ready to receive the first byte on a link within two processor cycles **TPCLPCL** after **Reset** goes low.

If the first byte received (the control byte) is greater than 1 it is taken as the quantity of bytes to be input. The following bytes, to that quantity, are then placed in internal memory starting at location *MemStart*. Following reception of the last byte the transputer will start executing code at *MemStart* as a low priority process. **BootFromROM** may be taken high after reception of the last byte, if required. The memory space immediately above the loaded code is used as work space. A byte arriving on other links after the control byte has been received and on the bootstrapping link after the last bootstrap byte, will be retained and no acknowledge will be sent until a process inputs from them.

4.6 Peek and poke

Any location in internal or external memory can be interrogated and altered when the transputer is waiting for a bootstrap from link. If the control byte is 0 then eight more bytes are expected on the same link. The first four byte word is taken as an internal or external memory address at which to poke (write) the second four byte word. If the control byte is 1 the next four bytes are used as the address from which to peek (read) a word of data; the word is sent down the output channel of the same link.

Following such a peek or poke, the transputer returns to its previously held state. Any number of accesses may be made in this way until the control byte is greater than 1, when the transputer will commence reading its bootstrap program. Any link can be used, but addresses and data must be transmitted via the same link as the control byte.

4.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initialises the transputer, triggers the memory configuration sequence and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) must not occur whilst **Reset** is high but can occur before bootstrap (page 101). After the end of **Reset** there will be a delay of 144 periods of **ClockIn** (figure 4.3). Following this, the **MemWrD0**, **MemRfD1** and **MemAD2-31** pins will be scanned to check for the existence of a pre-programmed memory interface configuration (page 104). This lasts for a further 144 periods of **ClockIn**. Regardless of whether a configuration was found, 36 configuration read cycles will then be performed on external memory using the default memory configuration (page 106), in an attempt to access the external configuration ROM. A delay will then occur, its period depending on the actual configuration. Finally eight complete and consecutive refresh cycles will initialise any dynamic RAM, using the new memory configuration. If the memory configuration does not enable refresh of dynamic RAM the refresh cycles will be replaced by an equivalent delay with no external memory activity.

If **BootFromROM** is high bootstrapping will then take place immediately, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

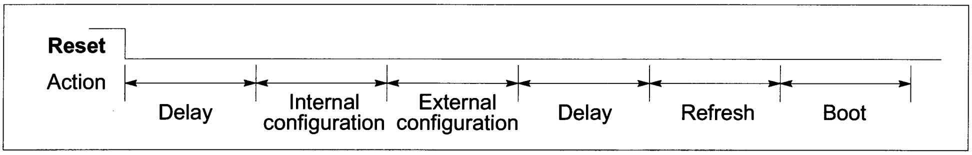


Figure 4.3 IMS T805 post-reset sequence

4.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 48). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags *Error*, *HaltOnError* and *EnableJ0Break* are normally cleared at reset on the IMS T805; however, if **Analyse** is asserted the flags are not altered. Memory refresh continues.

Input links will continue with outstanding transfers. Output links will not make another access to memory for data but will transmit only those bytes already in the link buffer. Providing there is no delay in link acknowledgement, the links should be inactive within a few microseconds of the transputer halting.

Reset should not be asserted before the transputer has halted and link transfers have ceased. When **Reset** is taken low whilst **Analyse** is high, neither the memory configuration sequence nor the block of eight refresh cycles will occur; the previous memory configuration will be used for any external memory accesses. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 4.3.

I	**MemStart** if bootstrapping from a link, or the external memory bootstrap address if bootstrapping from ROM.
W	**MemStart** if bootstrapping from ROM, or the address of the first free word after the bootstrap program if bootstrapping from link.
A	The value of **I** when the processor halted.
B	The value of **W** when the processor halted, together with the priority of the process when the transputer was halted (i.e. the **W** descriptor).
C	The ID of the bootstrapping link if bootstrapping from link.

Table 4.3 Register values after Analyse

Symbol	Parameter	T805-20, -25, -30			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

1 Full periods of **ClockIn TDCLDCL** required.

2 At power-on reset.

3 Must be stable until after end of bootstrap period. See Bootstrap section 4.5.

Table 4.4 **Reset** , **Analyse** and **BootFromROM** timing

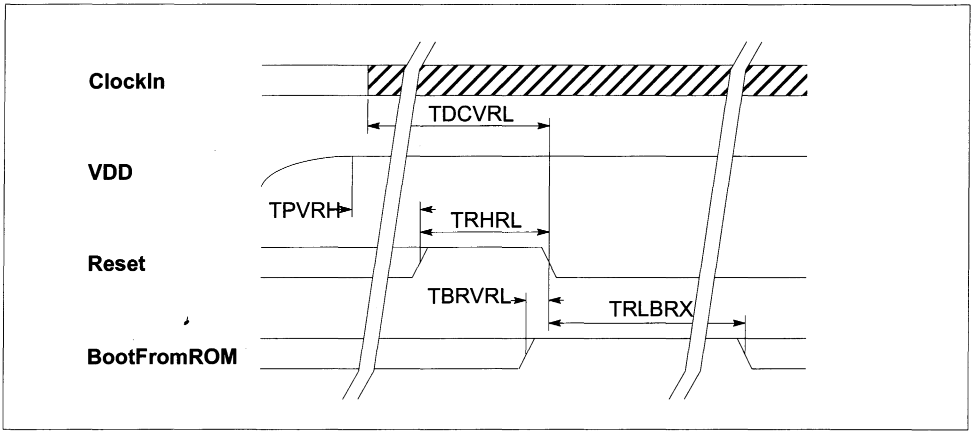


Figure 4.4 Transputer **Reset** timing with **Analyse** low

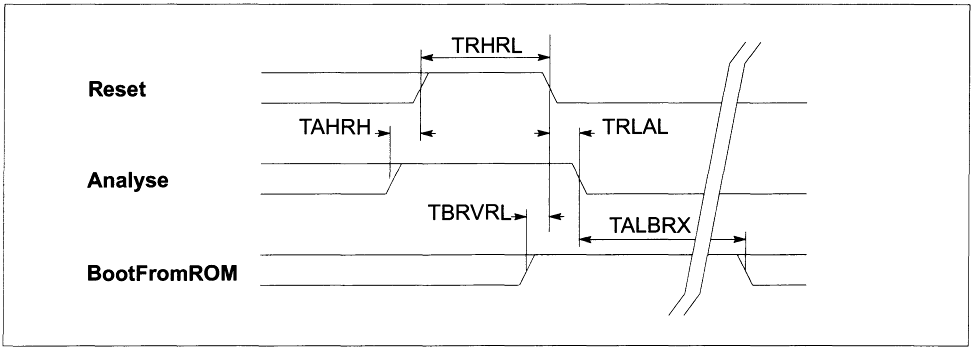


Figure 4.5 Transputer **Reset**, **Analyse** and **BootFromROM** timing

4.9 Error, ErrorIn

The **Error** pin carries the OR'ed output of the internal *Error* flag and the **ErrorIn** input. If **Error** is high it indicates either that **ErrorIn** is high or that an error was detected in one of the processes. An internal error can be caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 48). It can also be set from the floating point unit under certain circumstances (page 49, 76). Once set, the *Error* flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (page 81). A process can be programmed to stop if the *Error* flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data. **ErrorIn** does not directly affect the status of a processor in any way.

By setting the *HaltOnError* flag the transputer itself can be programmed to halt if *Error* becomes set. If *Error* becomes set after *HaltOnError* has been set, all processes on that transputer will cease but will not necessarily cause other transputers in a network to halt. Setting *HaltOnError* after *Error* will not cause the transputer to halt; this allows the processor reset and analyse facilities to function with the flags in indeterminate states.

An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by 'daisy-chaining' the **ErrorIn** and **Error** pins of a number of processors and applying the final **Error** output signal to the **EventReq** pin of a suitably programmed master transputer. Since the process state is preserved when stopped by an error, the master transputer can then use the analyse function to debug the fault. When using such a circuit, note that the *Error* flag is in an indeterminate state on power up; the circuit and software should be designed with this in mind.

Error checks can be removed completely to optimise the performance of a proven program; any unexpected error then occurring will have an arbitrary undefined effect.

If a high priority process pre-empts a low priority one, status of the *Error* and *HaltOnError* flags is saved for the duration of the high priority process and restored at the conclusion of it. Status of both flags is transmitted to the high priority process. Either flag can be altered in the process without upsetting the error status of any complex operation being carried out by the pre-empted low priority process.

In the event of a transputer halting because of *HaltOnError*, the links will finish outstanding transfers before shutting down. If **Analyse** is asserted then all inputs continue but outputs will not make another access to memory for data. Memory refresh will continue to take place.

After halting due to the *Error* flag changing from 0 to 1 whilst *HaltOnError* is set, register **I** points two bytes past the instruction which set *Error*. After halting due to the **Analyse** pin being taken high, register **I** points one byte past the instruction being executed. In both cases **I** will be copied to register **A**.

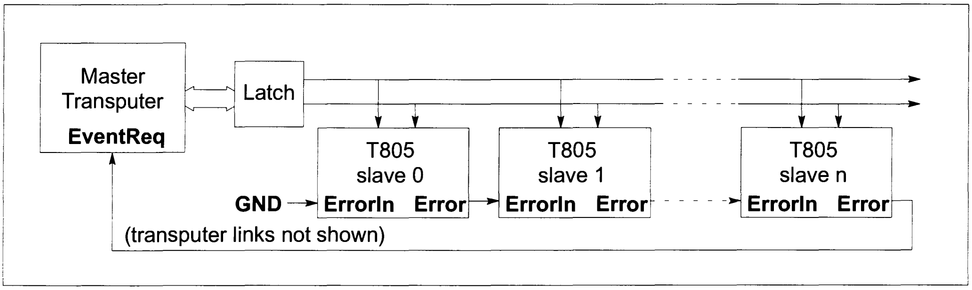


Figure 4.6 Error handling in a multi-transputer system

5 Memory

The IMS T805 has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (page 88). The transputer can also access 4 Gbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

IMS T805 memory is byte addressed, with words aligned on four-byte boundaries. The least significant byte of a word is the lowest addressed byte.

The bits in a byte are numbered 0 to 7, with bit 0 the least significant. The bytes are numbered from 0, with byte 0 the least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

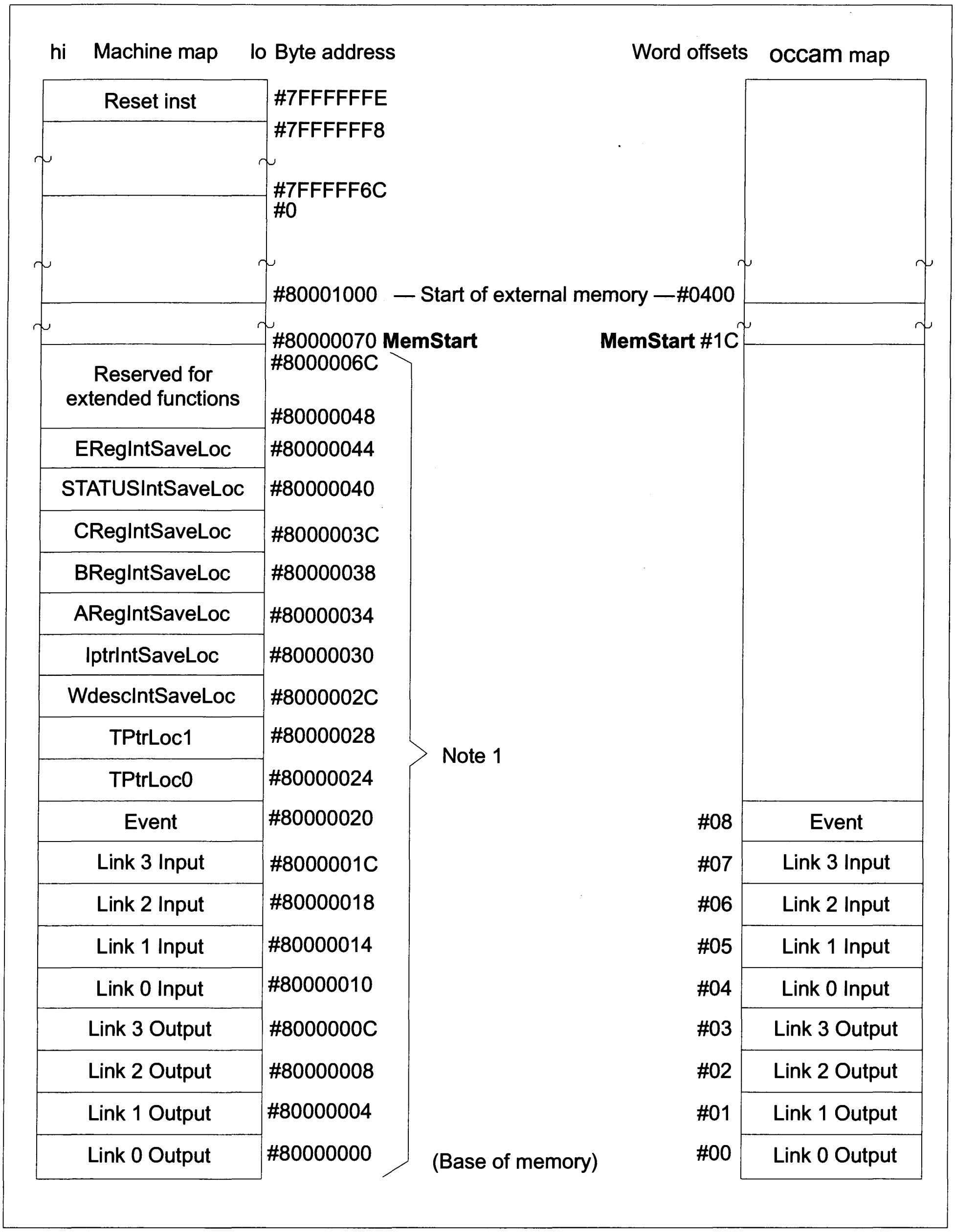
Internal memory starts at the most negative address #80000000 and extends to #80000FFF. User memory begins at #80000070; this location is given the name *MemStart*. An instruction *ldmemstartval* is provided to obtain the value of **MemStart**.

The context of a process in the transputer model involves a workspace descriptor (**WPtr**) and an instruction pointer (**IPtr**). **WPtr** is a word address pointer to a workspace in memory. **IPtr** points to the next instruction to be executed for the process which is the currently executing process. The context switch performed by the breakpoint instruction swaps the **WPtr** and **IPtr** of the currently executing process with the **WPtr** and **IPtr** held above **MemStart**. Two contexts are held above **MemStart**, one for high priority and one for low priority; this allows processes at both levels to have breakpoints. Note that on bootstrapping from a link, these contexts are overwritten by the loaded code. If this is not acceptable, the values should be peeked from memory before bootstrapping from a link. The reserved area of internal memory below **MemStart** is used to implement link and event channels.

Two words of memory are reserved for timer use, *TPtrLoc0* for high priority processes and *TPtrLoc1* for low priority processes. They either indicate the relevant priority timer is not in use or point to the first process on the timer queue at that priority level.

Values of certain processor registers for the current low priority process are saved in the reserved *IntSaveLoc* locations when a high priority process pre-empts a low priority one. Other locations are reserved for extended features such as block moves and floating point operations.

External memory space starts at #80001000 and extends up through #00000000 to #7FFFFFFF. Memory configuration data and ROM bootstrapping code must be in the most positive address space, starting at #7FFFFF6C and #7FFFFFFE respectively. Address space immediately below this is conventionally used for ROM based code.



hi Machine map	lo Byte address	Word offsets	occam map
Reset inst	#7FFFFFFE		
	#7FFFFFF8		
	#7FFFFF6C #0		
	#80001000 — Start of external memory —	#0400	
	#80000070 **MemStart**	**MemStart** #1C	
Reserved for extended functions	#8000006C #80000048		
ERegIntSaveLoc	#80000044		
STATUSIntSaveLoc	#80000040		
CRegIntSaveLoc	#8000003C		
BRegIntSaveLoc	#80000038		
ARegIntSaveLoc	#80000034		
IptrIntSaveLoc	#80000030		
WdescIntSaveLoc	#8000002C		
TPtrLoc1	#80000028		
TPtrLoc0	#80000024		
Event	#80000020	#08	Event
Link 3 Input	#8000001C	#07	Link 3 Input
Link 2 Input	#80000018	#06	Link 2 Input
Link 1 Input	#80000014	#05	Link 1 Input
Link 0 Input	#80000010	#04	Link 0 Input
Link 3 Output	#8000000C	#03	Link 3 Output
Link 2 Output	#80000008	#02	Link 2 Output
Link 1 Output	#80000004	#01	Link 1 Output
Link 0 Output	#80000000 (Base of memory)	#00	Link 0 Output

Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 81). For details see *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 5.1 IMS T805 memory map

6 External memory interface

The External Memory Interface (EMI) allows access to a 32 bit address space, supporting dynamic and static RAM as well as ROM and EPROM. EMI timing can be configured at **Reset** to cater for most memory types and speeds, and a program is supplied with the Transputer Development System to aid in this configuration.

There are 17 internal configurations which can be selected by a single pin connection (page 104). If none are suitable the user can configure the interface to specific requirements, as shown in page 106.

The external memory cycle is divided into six **Tstates** with the following functions:

T1 Address setup time before address valid strobe.

T2 Address hold time after address valid strobe.

T3 Read cycle tristate or write cycle data setup.

T4 Extendable data setup time.

T5 Read or write data.

T6 Data hold.

Under normal conditions each **Tstate** may be from one to four periods **Tm** long, the duration being set during memory configuration. The default condition on **Reset** is that all **Tstates** are the maximum four periods **Tm** long to allow external initialisation cycles to read slow ROM.

Period **T4** can be extended indefinitely by adding externally generated wait states.

An external memory cycle is always an even number of periods **Tm** in length and the start of **T1** always coincides with a rising edge of **ProcClockOut**. If the total configured quantity of periods **Tm** is an odd number, one extra period **Tm** will be added at the end of **T6** to force the start of the next **T1** to coincide with a rising edge of **ProcClockOut**. This period is designated **E** in configuration diagrams (figure 6.19).

During an internal memory access cycle the external memory interface bus **MemAD2-31** reflects the word address used to access internal RAM, **MemnotWrD0** reflects the read/write operation and **MemnotRfD1** is high; all control strobes are inactive. This is true unless and until a memory refresh cycle or DMA (memory request) activity takes place, when the bus will carry the appropriate external address or data.

The bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 80).

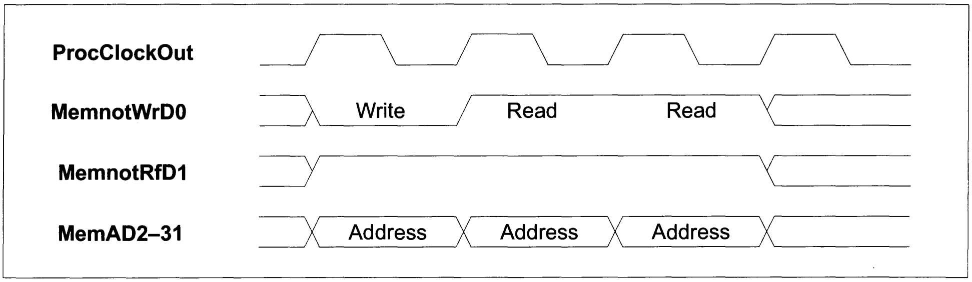


Figure 6.1 IMS T805 bus activity for internal memory cycle

6.1 Pin functions

6.1.1 MemAD2–31

External memory addresses and data are multiplexed on one bus. Only the top 30 bits of address are output on the external memory interface, using pins **MemAD2-31**. They are normally output only during **Tstates T1** and **T2**, and should be latched during this time. The data bus is 32 bits wide. It uses **MemAD2-31** for the top 30 bits and **MemnotRfD1** and **MemnotWrD0** for the lower two bits.

6.1.2 notMemRd

For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Data is read by the transputer on the rising edge of this strobe, and may be removed immediately afterward. If the strobe duration is insufficient it may be extended by adding extra periods **Tm** to either or both of the **Tstates T4** and **T5**. Further extension may be obtained by inserting wait states at the end of **T4**.

6.1.3 MemnotWrD0

During **T1** and **T2** this pin will be low if the cycle is a write cycle, otherwise it will be high. During **Tstates T3** to **T6** it becomes bit 0 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

6.1.4 notMemWrB0–3

Because the transputer uses word addressing, four write strobes are provided; one to write each byte of the word. **notMemWrB0** addresses the least significant byte.

6.1.5 notMemS0–4

To facilitate control of different types of memory and devices, the EMI is provided with five strobe outputs, four of which can be configured by the user. The strobes are conventionally assigned the functions shown in the read and write cycle diagrams, although there is no compulsion to retain these designations.

6.1.6 MemWait

Wait states can be selected by taking **MemWait** high. Externally generated wait states can be added to extend the duration of **T4** indefinitely.

6.1.7 MemnotRfD1

During **T1** and **T2**, this pin is low if the address on **MemAD2-31** is a refresh address, otherwise it is high. During **Tstates T3** to **T6** it becomes bit 1 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

6.1.8 notMemRf

The IMS T805 can be operated with memory refresh enabled or disabled. The selection is made during memory configuration, when the refresh interval is also determined.

6.1.9 RefreshPending

When high, this pin signals that a refresh cycle is pending.

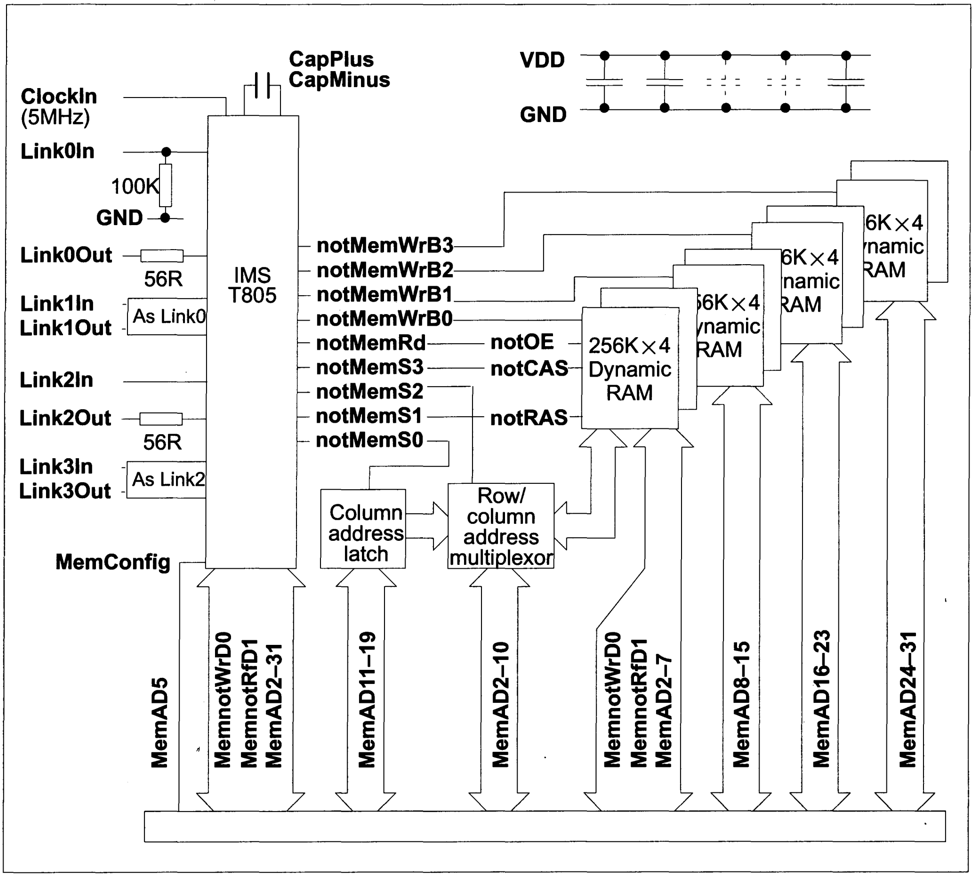


Figure 6.2 IMS T805 application

6.1.10 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high. **MemGranted** follows the timing of the bus being tristated and can be used to signal to the device requesting the DMA that it has control of the bus. Note that **MemGranted** changes on the falling edge of **ProcClockOut** and can therefore be sampled to establish control of the bus on the rising edge of **ProcClockOut**.

6.1.11 MemConfig

MemConfig is an input pin used to read configuration data when setting external memory interface (EMI) characteristics.

6.1.12 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn**. Its period is equal to one internal microcode cycle time, and can be derived from the formula

TPCLPCL = TDCLDCL / PLLx

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (Ordering section).

The time value **Tm** is used to define the duration of **Tstates** and, hence, the length of external memory cycles; its value is exactly half the period of one **ProcClockOut** cycle (0.5***TPCLPCL**), regardless of mark/ space ratio of **ProcClockOut**.

Edges of the various external memory strobes coincide with rising or falling edges of **ProcClockOut**. It should be noted, however, that there is a skew associated with each coincidence. The value of skew depends on whether coincidence occurs when the **ProcClockOut** edge and strobe edge are both rising, when both are falling or if either is rising when the other is falling. Timing values given in the strobe tables show the best and worst cases. If a more accurate timing relationship is required, the exact **Tstate** timing and strobe edge to **ProcClockOut** relationships should be calculated and the correct skew factors applied from the edge skew timing table 6.4.

		T805-20		**T805-25**		**T805-30**			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TPCLPCL	**ProcClockOut** period	48	52	38	42	31.5	35	ns	2
TPCHPCL	**ProcClockOut** pulse width high	13.5	28.5	8.5	23.5	13	22	ns	2
TPCLPCH	**ProcClockOut** pulse width low	**a**		**a**		**a**		ns	2, 3, 4
Tm	**ProcClockOut** half cycle	24	26	19	21	15.5	17.5	ns	2
TPCstab	**ProcClockOut** stability		8		8		8	%	1,2

Notes

1 Stability is the variation of cycle periods between two consecutive cycles, measured at corresponding points on the cycles.

2 This parameter is sampled and not 100% tested.

3 **a** is TPCLPCL – TPCHPCL.

4 This is a nominal value.

Table 6.1 **ProcClockOut**

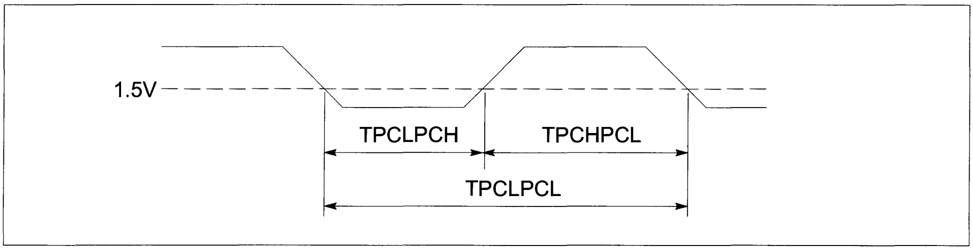


Figure 6.3 IMS T805 **ProcClockOut** timing

6.2 Read cycle

Byte addressing is carried out internally by the transputer for read cycles. For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Read cycle data may be set up on the data bus at any time after the start of **T3**, but must be valid when the transputer reads it at the end of **T5**. Data may be removed any time during **T6**, but must be off the bus no later than, the end of that period.

notMemS0 is a fixed format strobe. Its leading edge is always coincident with the start of **T2** and its trailing edge always coincident with the end of **T5**.

The leading edge of **notMemS1** is always coincident with the start of **T2**, but its duration may be configured to be from zero to 31 periods **Tm**. Regardless of the configured duration, the strobe will terminate no later than the end of **T6**. The strobe is sometimes programmed to extend beyond the normal end of **Tmx**. When wait states are inserted into an EMI cycle the end of **Tmx** is delayed, but the potential active duration of the strobe is not altered. Thus the strobe can be configured to terminate relatively early under certain conditions (page 96). If **notMemS1** is configured to be zero it will never go low.

notMemS2, **notMemS3** and **notMemS4** are identical in operation. They all terminate at the end of **T5**, but the start of each can be delayed from one to 31 periods **Tm** beyond the start of **T2**. If the duration of one of these strobes would take it past the end of **T5** it will stay high. This can be used to cause a strobe to become active only when wait states are inserted. If one of these strobes is configured to zero it will never go low. Figure 6.6 shows the effect of **Wait** on strobes in more detail; each division on the scale is one period **Tm**.

In the read cycle timing diagrams **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**.

		T805-20		T805-25		T805-30			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Note**
TaZdV	Address tristate to data valid	0		0		0		ns	3
TdVRdH	Data setup before read	25		20		15		ns	
TRdHdX	Data hold after read	0		0		0		ns	3
TS0LRdL	notMemS0 before start of read	**a**–4	**a**+4	**a**–4	**a**+4	**a**–3	**a**+3	ns	1
TS0HRdH	End of read from end of notMemS0	–4	4	–4	4	–4	2	ns	
TRdLRdH	Read period	**b**–3	**b**+5	**b**–3	**b**+5	**b**–3	**b**+3	ns	2

Notes

1 **a** is total of **T2**+**T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

2 **b** is total of **T4**+**Twait**+**T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

3 Guaranteed, but not tested.

Table 6.2 Read

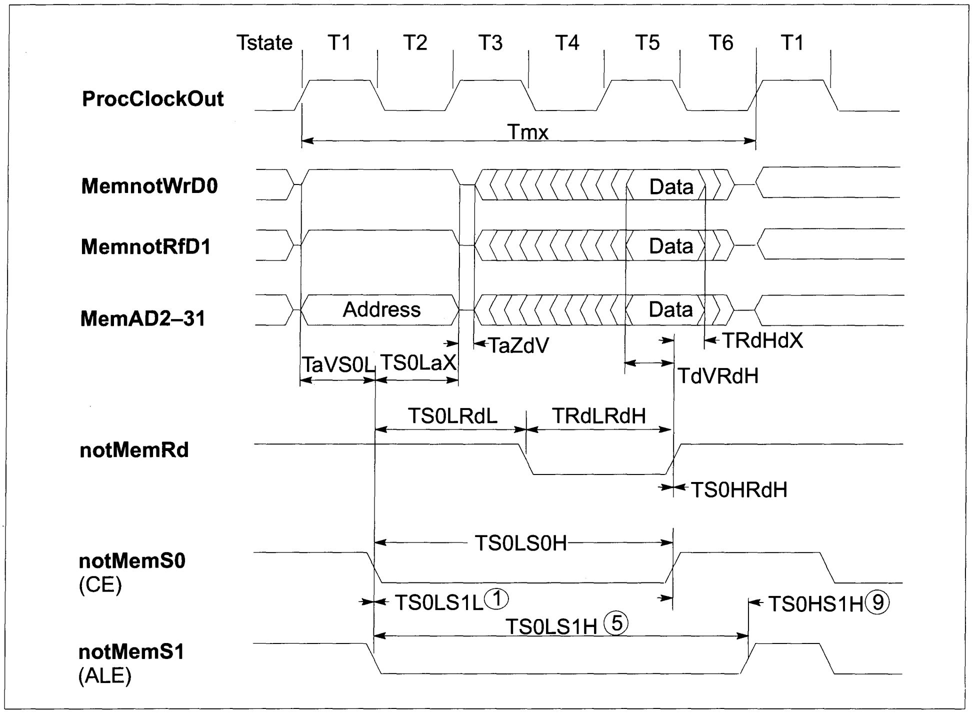


Figure 6.4 IMS T805 external read cycle: static memory

Symbol	n	Parameter	T805-20		T805-25		T805-30		Note
			Min	Max	Min	Max	Min	Max	
TaVS0L		Address setup before **notMemS0**	**a**–8		**a**–8		**a**–4		1
TS0LaX		Address hold after **notMemS0**	**b**–8	**b**+8	**b**–8	**b**+8	**b**–4	**b**+5	2
TS0LS0H		**notMemS0** pulse width low	**c**–5	**c**+6	**c**–5	**c**+6	**c**–5	**c**+6	3
TS0LS1L	1	**notMemS1** from **notMemS0**	–4	4	–4	4	–4	4	10
TS0LS1H	5	**notMemS1** end from **notMemS0**	**d**–1	**d**+9	**d**–1	**d**+9	**d**–1	**d**+5	4,6
TS0HS1H	9	**notMemS1** end from **notMemS0** end	**e**–8	**e**+4	**e**–8	**e**+4	**e**–5	**e**+2	5,6
TS0LS2L	2	**notMemS2** delayed after **notMemS0**	**f**–6	**f**+5	**f**–6	**f**+5	**f**–4	**f**+3	7
TS0LS2H	6	**notMemS2** end from **notMemS0**	**c**–5	**c**+7	**c**–5	**c**+7	**c**–5	**c**+3	3,10
TS0HS2H	10	**notMemS2** end from **notMemS0** end	–4	7	–4	7	–4	5	10
TS0LS3L	3	**notMemS3** delayed after **notMemS0**	**f**–6	**f**+5	**f**–6	**f**+5	**f**–4	**f**+3	7
TS0LS3H	7	**notMemS3** end from **notMemS0**	**c**–5	**c**+7	**c**–5	**c**+7	**c**–5	**c**+3	3,10
TS0HS3H	11	**notMemS3** end from **notMemS0** end	–4	7	–4	7	–4	5	10
TS0LS4L	4	**notMemS4** delayed after **notMemS0**	**f**–6	**f**+5	**f**–6	**f**+5	**f**–4	**f**+3	7
TS0LS4H	8	**notMemS4** end from **notMemS0**	**c**–5	**c**+7	**c**–5	**c**+7	**c**–5	**c**+3	3,10
TS0HS4H	12	**notMemS4** end from **notMemS0** end	–4	7	–4	7	–4	5	10
Tmx		Complete external memory cycle							8,9
All timings in nanoseconds (ns).									

Notes

1 **a** is **T1** where **T1** can be from one to four periods **Tm** in length.

2 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

3 **c** is total of **T2+T3+T4+Twait+T5** where **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

4 **d** can be from zero to 31 periods **Tm** in length.

5 **e** can be from –27 to +4 periods **Tm** in length.

6 If the configuration would cause the strobe to remain active past the end of **T6** it will go high at the end of **T6**. If the strobe is configured to zero periods **Tm** it will remain high throughout the complete cycle **Tmx**.

7 **f** can be from zero to 31 periods **Tm** in length. If this length would cause the strobe to remain active past the end of **T5** it will go high at the end of **T5**. If the strobe value is zero periods **Tm** it will remain low throughout the complete cycle **T1** to **T5**, going high only for first **Tm** of **T6**.

8 **Tmx** is one complete external memory cycle comprising the total of **T1+T2+T3+T4+Twait+T5+T6** where **T1**, **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length, **T6** can be from one to five periods **Tm** in length and **Twait** may be zero or any number of periods **Tm** in length.

9 Guaranteed, but not tested.

10 Sampled, not 100% tested.

Table 6.3 IMS T805 strobe timing

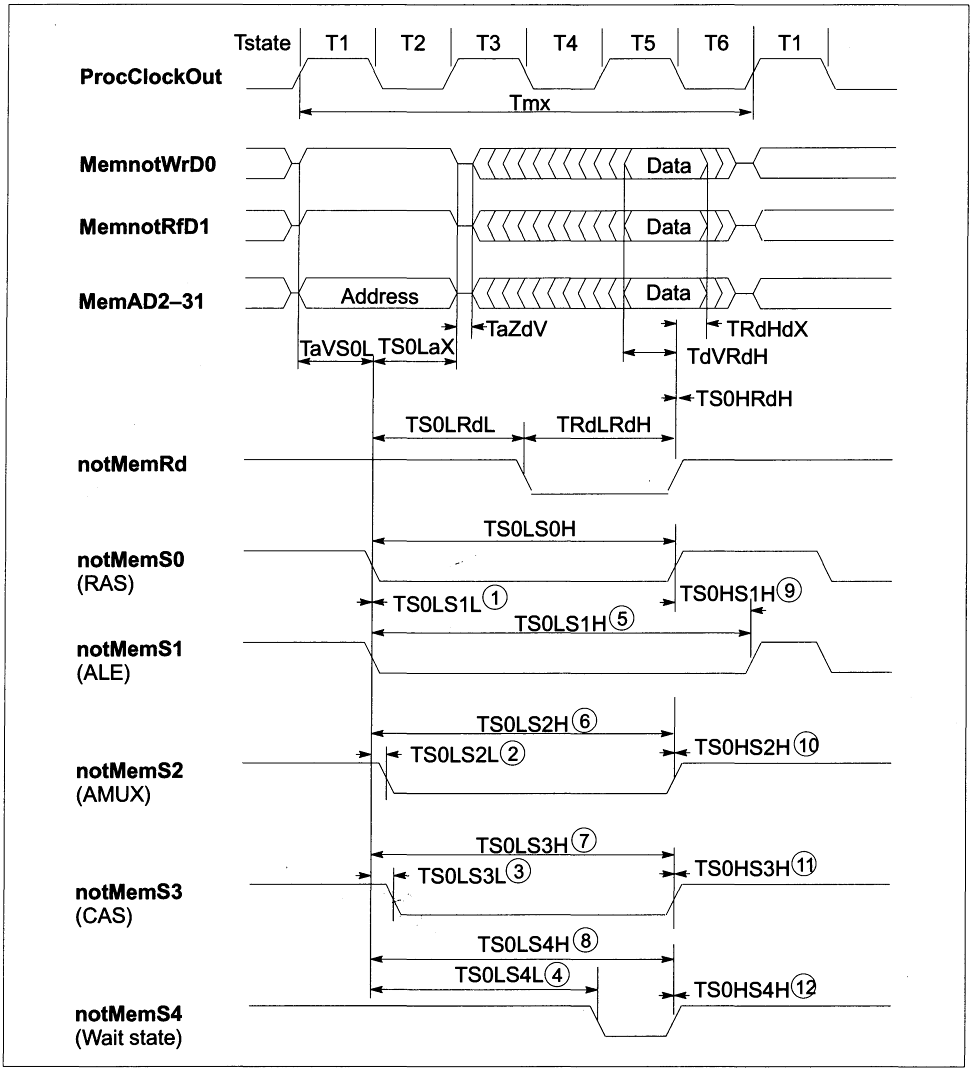


Figure 6.5 IMS T805 external read cycle: dynamic memory

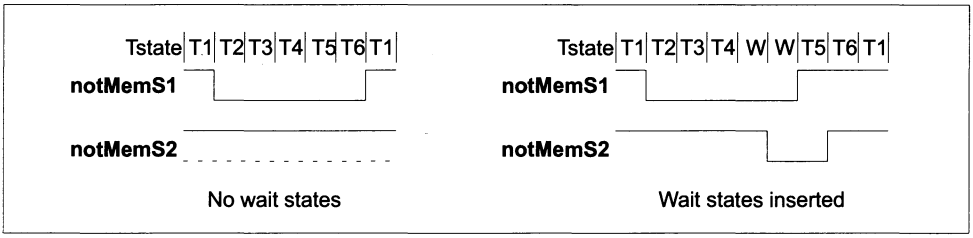


Figure 6.6 IMS T805 effect of wait states on strobes

Symbol	Parameter	T805-20		T805-25		T805-30		Note
		Min	Max	Min	Max	Min	Max	
TPCHS0H	notMemS0 rising from ProcClockOut rising	–6	4	–6	4	–4	3	1
TPCLS0H	notMemS0 rising from ProcClockOut falling	–5	10	–5	10	–3.5	7	1
TPCHS0L	notMemS0 falling from ProcClockOut rising	–8	3	–8	3	–5.5	2	1
TPCLS0L	notMemS0 falling from ProcClockOut falling	–5	7	–5	7	–3.5	5	1
All timings in nanoseconds (ns).								

Notes

1 Sampled, not 100% tested.

Table 6.4 Strobe S0 to **ProcClockOut** skew

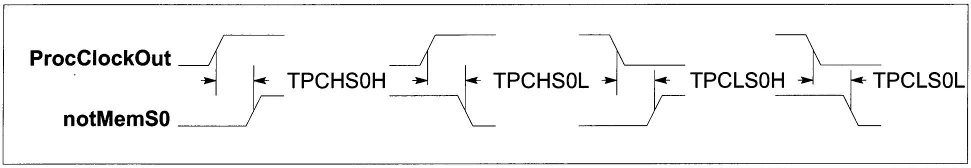


Figure 6.7 IMS T805 skew of **notMemS0** to **ProcClockOut**

6.3 Write cycle

For write cycles the relevant bytes in memory are addressed by the write strobes **notMemWrB0-3**. If a particular byte is not to be written, then the corresponding data outputs are tristated.

For a write cycle pin **MemnotWrD0** will be low during **T1** and **T2**. Write data is placed on the bus at the start of **T3** and removed at the end of **T6**. If **T6** is extended to force the next cycle **Tmx** (page 87) to start on a rising edge of **ProcClockOut**, data will be valid during this time also.

The transputer has both early and late write cycle modes. For a late write cycle the relevant write strobes **notMemWrB0-3** are low during **T4** and **T5**; for an early write they are also low during **T3**. Data should be latched into memory on the rising edge of the strobes in both cases, although it is valid until the end of **T6**. If the strobe duration is insufficient, it may be extended at configuration time by adding extra periods **Tm** to either or both of **Tstates T4** and **T5** for both early and late modes. For an early cycle they may also be added to **T3**. Further extension may be obtained by inserting wait states at the end of **T4**. If the data hold time is insufficient, extra periods **Tm** may be added to **T6** to extend it.

In the write cycle timing diagram **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**. The strobe is inactive during internal memory cycles.

Symbol	Parameter	T805-20		T805-25		T805-30		Notes
		Min	Max	Min	Max	Min	Max	
TdVWrH	Data setup before write	d–7	d+10	d–7	d+10	d–4	d+5	1,5
TWrHdX	Data hold after write	a–10	a+5	a–10	a+5	a–6	a+2	1,2
TS0LWrL	**notMemS0** before start of early write	b–5	b+5	b–5	b+5	b–5	b+2	1,3
	notMemS0 before start of late write	c–5	c+5	c–5	c+5	c–5	c+2	1,4
TS0HWrH	End of write from end of **notMemS0**	–5	4	–5	4	–5	4	1,7
TWrLWrH	Early write pulse width	d–4	d+7	d–4	d+7	d–4	d+4	1,5
	Late write pulse width	e–4	e+7	e–4	e+7	e–4	e+4	1,6
All timings in nanoseconds (ns).								

Notes

1 Timing is for all write strobes **notMemWrB0-3**.

2 **a** is **T6** where **T6** can be from one to five periods **Tm** in length.

3 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

4 **c** is total of **T2+T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

5 **d** is total of **T3+T4+Twait+T5** where **T3, T4, T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

6 **e** is total of **T4+Twait+T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

7 Sampled, not 100% tested.

Table 6.5 Write

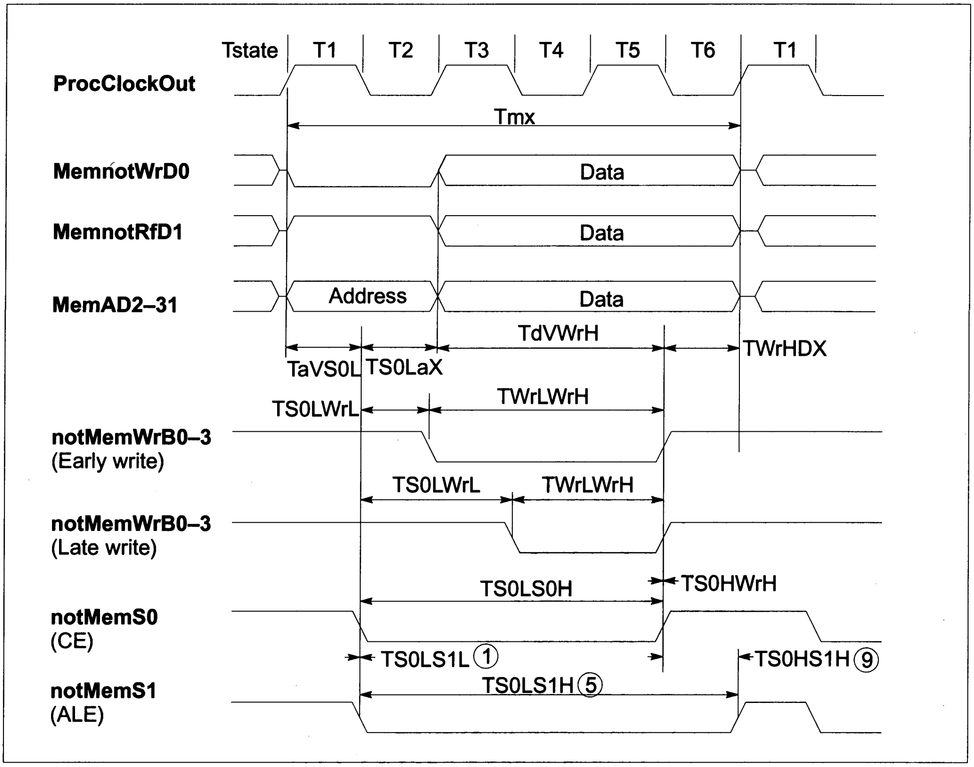


Figure 6.8 IMS T805 external write cycle

6.4 Wait

Taking **MemWait** high with the timing shown (figure 6.9) will extend the duration of **T4**. **MemWait** is sampled close to the falling edge of **ProcClockOut** prior to, but not at, the end of **T4**. By convention, **notMemS4** is used to synchronize wait state insertion. If this or another strobe is used, its delay should be such as to take the strobe low an even number of periods **Tm** after the start of **T1**, to coincide with a rising edge of **ProcClockOut**.

MemWait may be kept high indefinitely, although if dynamic memory refresh is used it should not be kept high long enough to interfere with refresh timing. **MemWait** operates normally during all cycles, including refresh and configuration cycles. It does not affect internal memory access in any way.

If the start of **T5** would coincide with a falling edge of **ProcClockOut** an extra wait period **Tm** (**EW**) is generated by the EMI to force coincidence with a rising edge. Rising edge coincidence is only forced if wait states are added, otherwise coincidence with a falling edge is permitted.

Symbol	Parameter	T805–20		T805–25		T805–30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TPCLWtH	Wait setup	10		10		8		ns	1,2
TPCLWtL	Wait hold	8		8		6		ns	1,2
TWtLWtH	Delay before re-assertion of Wait	50		40		33		ns	3

Notes

1 **ProcClockOut** load should not exceed 50pf.

2 If wait period exceeds refresh interval, refresh cycles will be lost.

3 Guaranteed, but not tested.

Table 6.6 IMS T805 memory wait

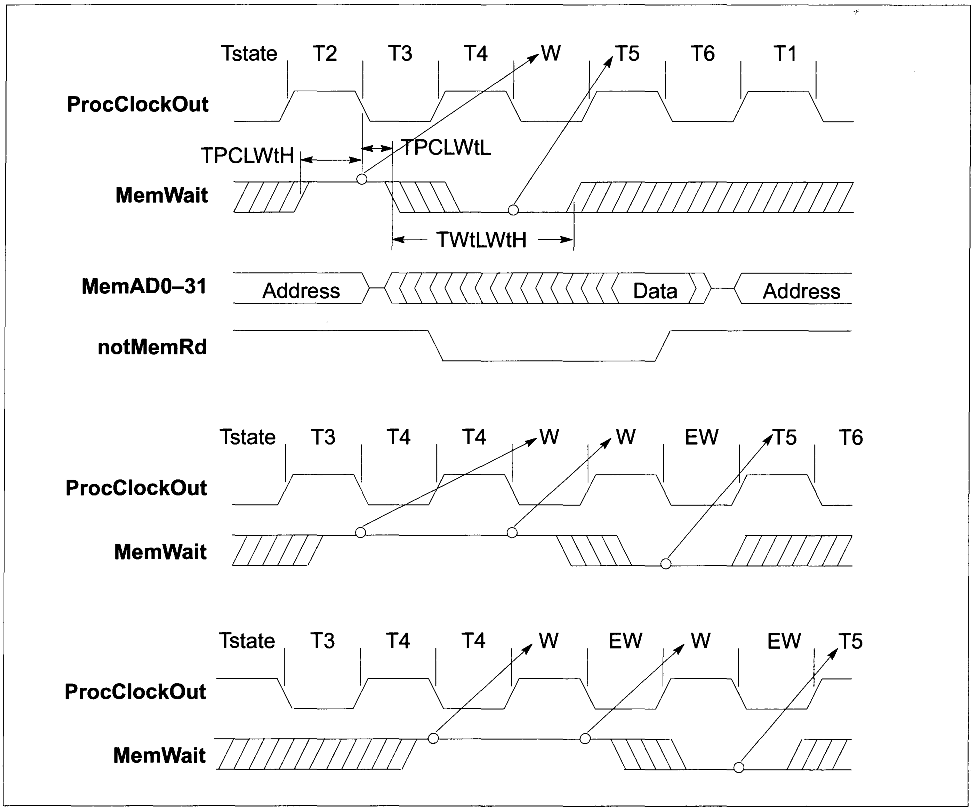


Figure 6.9 IMS T805 memory wait timing

6.5 Memory refresh

The **RefreshPending** pin is asserted high when the external memory interface is about to perform a refresh cycle. It remains high until the refresh cycle is started by the transputer. The minimum time for the **RefreshPending** pin to be high is for one cycle of **ProcClockOut** (two periods **Tm**), when the EMI was not about to perform a memory read or write. If the EMI was held in the tristate condition with **MemGranted** asserted, then **RefreshPending** will be asserted when the refresh controller in the EMI is ready to perform a refresh. **MemReq** may be re-asserted any time after the commencement of the refresh cycle. **RefreshPending** changes state near the rising edge of **ProcClockOut** and can therefore be sampled by the falling edge of **ProcClockOut**.

If no DMA is active then refresh will be performed following the end of the current internal or external memory cycle. If DMA is active the transputer will wait for DMA to terminate before commencing the refresh cycle. Unlike **MemnotRfD1**, **RefreshPending** is never tristated and can thus be interrogated by the DMA device; the DMA cycle can then be suspended, at the discretion of the DMA device, to allow refresh to take place.

The simple circuit of Figure 6.10 will suspend DMA requests from the external logic when **RefreshPending** is asserted, so that a memory refresh cycle can be performed. DMA is restored on completion of the refresh cycle. The transputer will not perform an external memory cycle other than a refresh cycle, using this method, until the requesting device removes its DMA request.

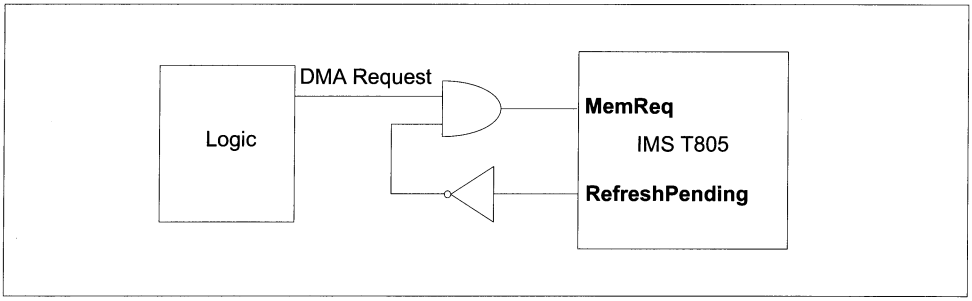


Figure 6.10 IMS T805 refresh with DMA

When refresh is disabled no refresh cycles occur. During the post-**Reset** period eight dummy refresh cycles will occur with the appropriate timing but with no bus or strobe activity.

A refresh cycle uses the same basic external memory timing as a normal external memory cycle, except that it starts two periods **Tm** before the start of **T1**. If a refresh cycle is due during an external memory access, it will be delayed until the end of that external cycle. Two extra periods **Tm** (periods **R** in the diagram) will then be inserted between the end of **T6** of the external memory cycle and the start of **T1** of the refresh cycle itself. The refresh address and various external strobes become active approximately one period **Tm** before **T1**. Bus signals are active until the end of **T2**, whilst **notMemRf** remains active until the end of **T6**.

For a refresh cycle, **MemnotRfD1** goes low before **notMemRf** goes low and **MemnotWrD0** goes high with the same timing as **MemnotRfD1**. All the address lines share the same timing, but only **MemAD2-11** give the refresh address. **MemAD12-30** stay high during the address period, whilst **MemAD31** remains low. Refresh cycles generate strobes **notMemS0-4** with timing as for a normal external cycle, but **notMemRd** and **notMemWrB0-3** remain high. **MemWait** operates normally during refresh cycles.

Refresh cycles do not interrupt internal memory accesses, although the internal addresses cannot be reflected on the external bus during refresh.

Symbol	Parameter	T805-20		T805-25		T805-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TRfLRfH	Refresh pulse width low	a–2	a+9	a–2	a+9	a–2	a+9	ns	1
TRaVS0L	Refresh address setup before **notMemS0**	b–12		b–12		b–12		ns	
TRfLS0L	Refresh indicator setup before **notMemS0**	b–4	b+6	b–4	b+6	b–4	b+6	ns	2

Notes

1 **a** is total **Tmx+Tm**.

2 **b** is total **T1+Tm** where **T1** can be from one to four periods **Tm** in length.

Table 6.7 Memory refresh

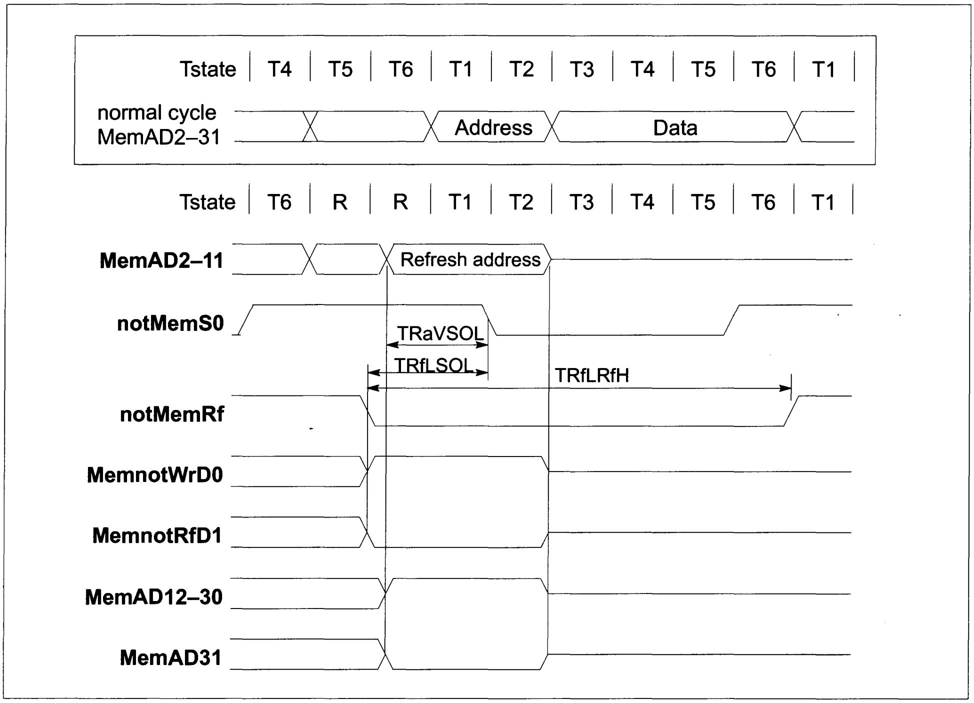


Figure 6.11 IMS T805 refresh cycle timing

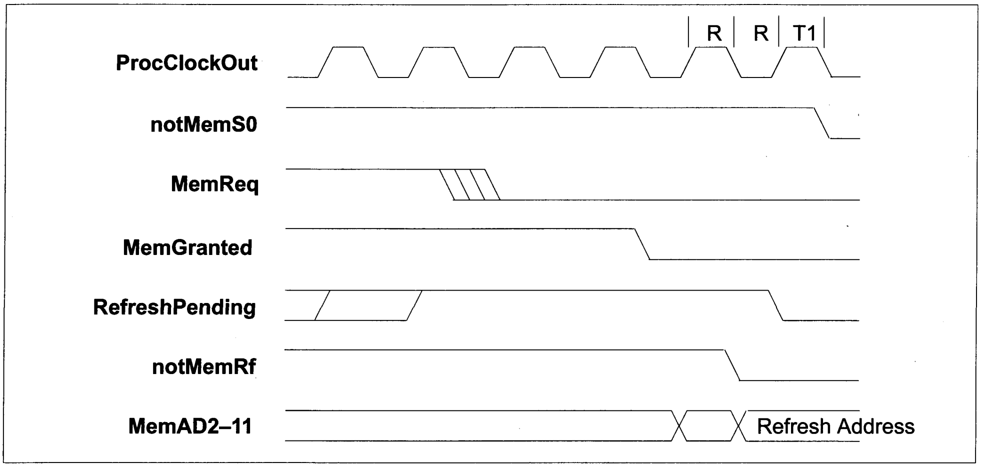


Figure 6.12 IMS T805 refresh pending timing diagram

6.6 Direct memory access

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high. The transputer samples **MemReq** just before falling edges of **ProcClockOut**. To guarantee taking over the bus immediately following either a refresh or external memory cycle, **MemReq** must be sampled at least four periods **Tm** before the end of **T6**. In the absence of an external memory cycle, the address bus is tristated two periods **Tm** after the **ProcClockOut** rising edge which follows the sample.

Removal of **MemReq** is sampled just before falling edges of **ProcClockOut** and **MemGranted** is removed synchronously with the second falling edge of **ProcClockOut** which follows the sample. If accurate timing of DMA is required, the setup time relative to **ProcClockOut** must be met. Further external bus activity, either refresh, external cycles or reflection of internal cycles, will commence at the next but one rising edge of **ProcClockOut**.

The strobes (**notMemS0–4** and **notMemWrB0–3**) are left in their inactive states during DMA. DMA cannot interrupt a refresh or external memory cycle, and outstanding refresh cycles will occur before the bus is released to DMA. DMA does not interfere with internal memory cycles in any way, although a program running in internal memory would have to wait for the end of DMA before accessing external memory. DMA cannot access internal memory. If DMA extends longer than one refresh interval (Memory Refresh Configuration Coding, table 6.11), the DMA user becomes responsible for refresh (see section 6.5). DMA may also inhibit an internally running program from accessing external memory.

DMA allows a bootstrap program to be loaded into external RAM ready for execution after reset. If **MemReq** is held high throughout reset, **MemGranted** will be asserted before the bootstrap sequence begins. **MemReq** must be high at least one period **TDCLDCL** of **ClockIn** before **Reset**. The circuit should be designed to ensure correct operation if **Reset** could interrupt a normal DMA cycle.

		T805–20		T805–25		T805–30			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Note**
TMRHPCL	**MemReq** setup before **ProcClockOut** falling	3	14	3	14	4	14	ns	1
TPCLMGH	**MemReq** response time	96	110	77	89	65	75	ns	2
TMRLPCL	**Memreq** removal before **ProcClockOut** falling	4	16	4	15	4	14		
TPCLMGL	**MemReq** end response time	50	66	40	54	37	49	ns	
TADZMGH	Bus tristate before **MemGranted**	0	27	0	22	0	18	ns	
TMGLADV	Bus active after end of **MemGranted**	0	32	0	26	0	20	ns	

Notes

1 Setup time need only be met to guarantee sampling on this edge.

2 If an external cycle is active, maximum time could be (1 EMI cycle **Tmx**)+(1 refresh cycle **TRfLRfH**)+(6 periods **Tm**).

Table 6.8 Memory request

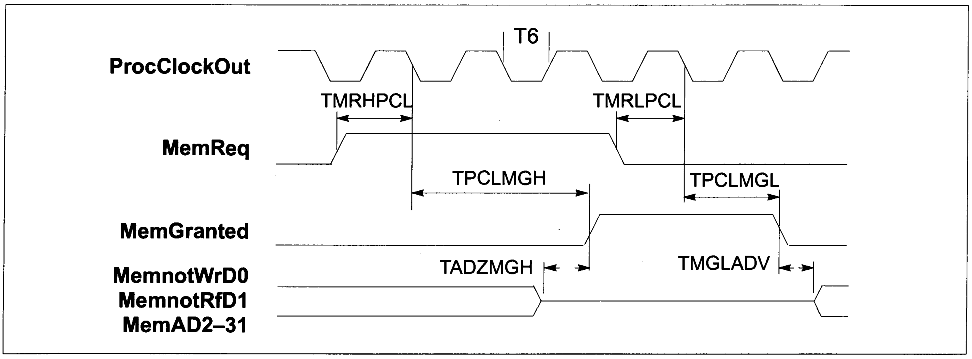
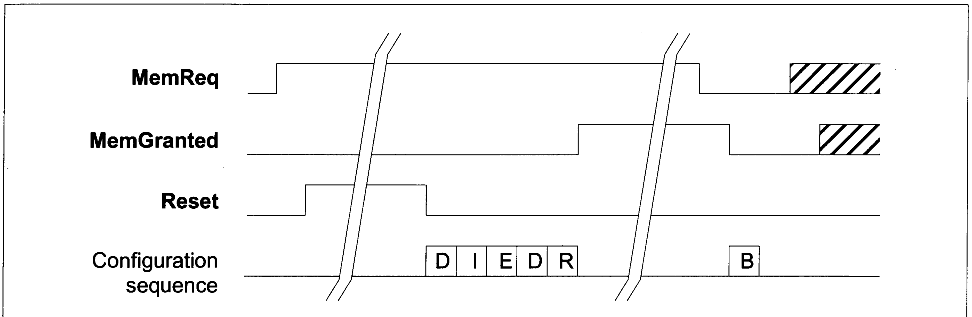


Figure 6.13 IMS T805 memory request timing



D Pre– and post–configuration delays (figure 4.3)
I Internal configuration sequence
E External configuration sequence
R Initial refresh sequence
B Bootstrap sequence

Figure 6.14 IMS T805 DMA sequence at reset

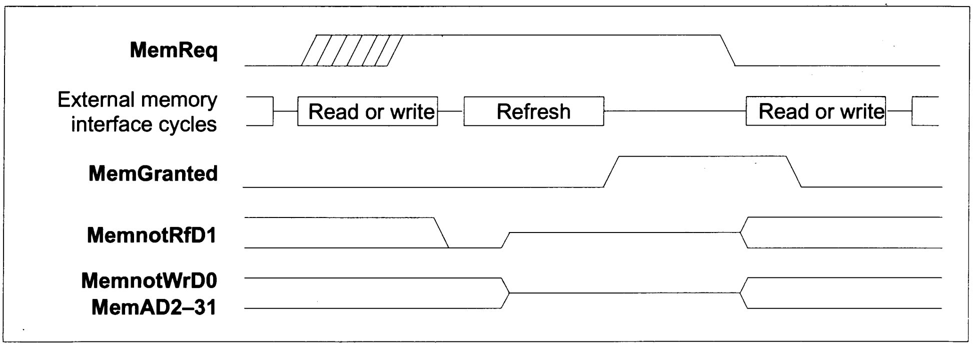


Figure 6.15 IMS T805 operation of **MemReq**, **MemGranted** with external, refresh memory cycles

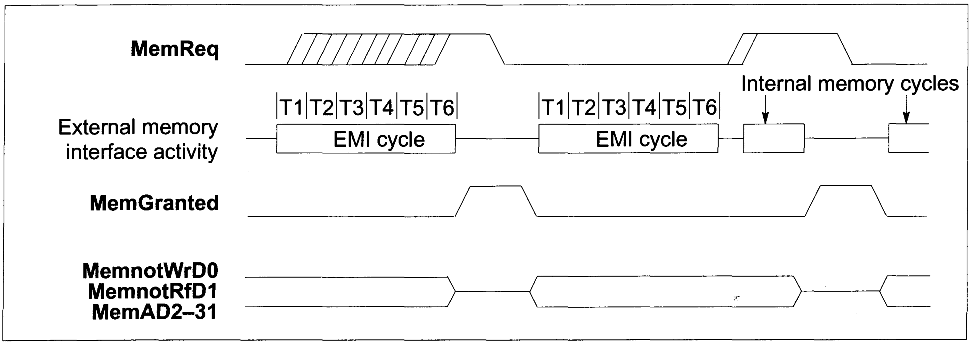


Figure 6.16 IMS T805 operation of **MemReq**, **MemGranted** with external, internal memory cycles

6.7 Memory configuration

MemConfig is an input pin used to read configuration data when setting external memory interface (EMI) characteristics. It is read by the processor on two occasions after **Reset** goes low; first to check if one of the preset internal configurations is required, then to determine a possible external configuration.

6.7.1 Internal configuration

The internal configuration scan comprises 64 periods **TDCLDCL** of **ClockIn** during the internal scan period of 144 **ClockIn** periods. **MemnotWrD0**, **MemnotRfD1** and **MemAD2-32** are all high at the beginning of the scan. Starting with **MemnotWrD0**, each of these lines goes low successively at intervals of two **ClockIn** periods and stays low until the end of the scan. If one of these lines is connected to **MemConfig** the preset internal configuration mode associated with that line will be used as the EMI configuration. The default configuration is that defined in the table for **MemAD31**; connecting **MemConfig** to **VDD** will also produce this default configuration. Note that only 17 of the possible configurations are valid, all others remain at the default configuration.

Pin	Duration of each Tstate periods Tm						Strobe coefficient				Write cycle	Refresh interval	Cycle time
	T1	T2	T3	T4	T5	T6	s1	s2	s3	s4	type	ClockIn cycles	Proc cycles
MemnotWrD0	1	1	1	1	1	1	30	1	3	5	late	72	3
MemnotRfD1	1	2	1	1	1	2	30	1	2	7	late	72	4
MemAD2	1	2	1	1	2	3	30	1	2	7	late	72	5
MemAD3	2	3	1	1	2	3	30	1	3	8	late	72	6
MemAD4	1	1	1	1	1	1	3	1	2	3	early	72	3
MemAD5	1	1	2	1	2	1	5	1	2	3	early	72	4
MemAD6	2	1	2	1	3	1	6	1	2	3	early	72	5
MemAD7	2	2	2	1	3	2	7	1	3	4	early	72	6
MemAD8	1	1	1	1	1	1	30	1	2	3	early	†	3
MemAD9	1	1	2	1	2	1	30	2	5	9	early	†	4
MemAD10	2	2	2	2	4	2	30	2	3	8	late	72	7
MemAD11	3	3	3	3	3	3	30	2	4	13	late	72	9
MemAD12	1	1	2	1	2	1	4	1	2	3	early	72	4
MemAD13	2	1	2	1	2	2	5	1	2	3	early	72	5
MemAD14	2	2	2	1	3	2	6	1	3	4	early	72	6
MemAD15	2	1	2	3	3	3	8	1	2	3	early	72	7
MemAD31	4	4	4	4	4	4	31	30	30	18	late	72	12

† Provided for static RAM only.

Table 6.9 IMS T805 internal configuration coding

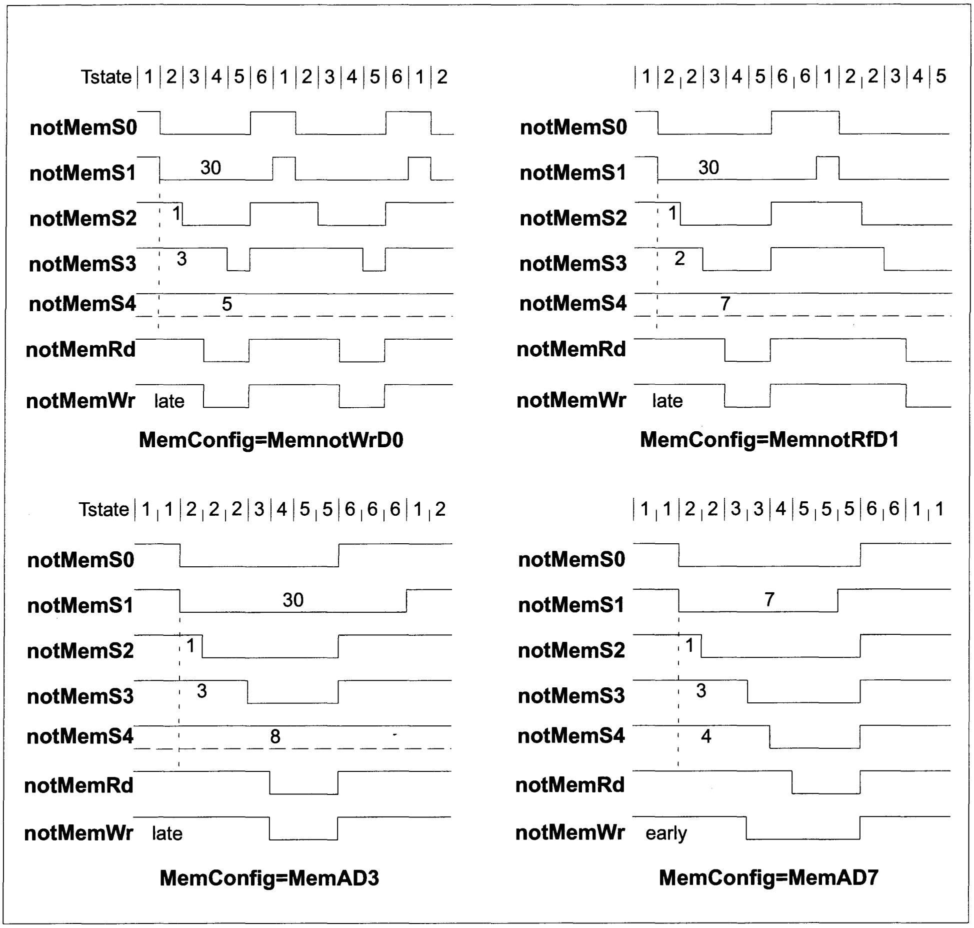
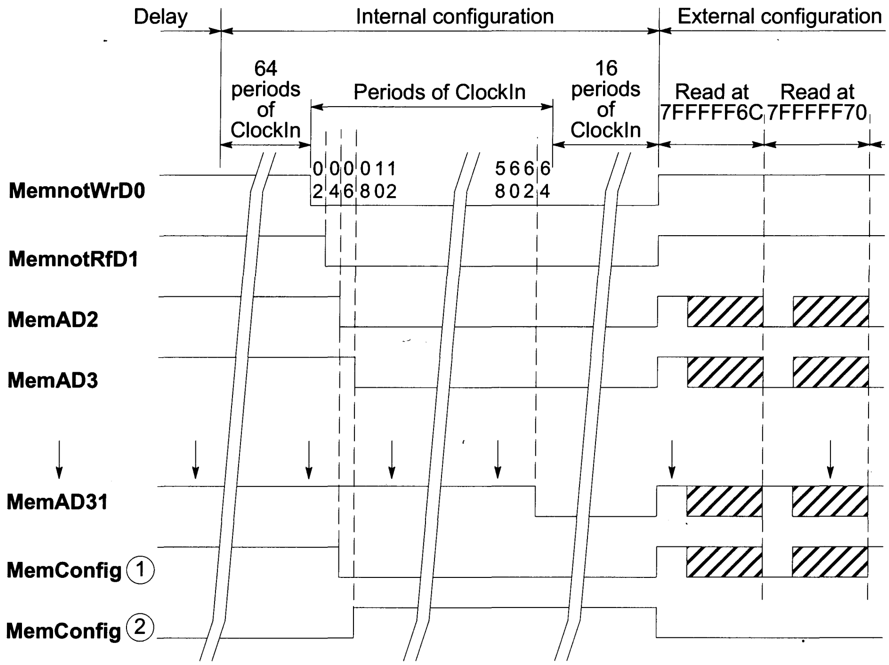


Figure 6.17 IMS T805 internal configuration



(1) Internal configuration: **MemConfig** connected to **MemAD2**

(2) External configuration: **MemConfig** connected to inverse of **MemAD3**

Figure 6.18 IMS T805 internal configuration scan

6.7.2 External configuration

If **MemConfig** is held low until **MemnotWrD0** goes low the internal configuration is ignored and an external configuration will be loaded instead. An external configuration scan always follows an internal one, but if an internal configuration occurs any external configuration is ignored.

The external configuration scan comprises 36 successive external read cycles, using the default EMI configuration preset by **MemAD31**. However, instead of data being read on the data bus as for a normal read cycle, only a single bit of data is read on **MemConfig** at each cycle. Addresses put out on the bus for each read cycle are shown in table 6.10, and are designed to address ROM at the top of the memory map. The table shows the data to be held in ROM; data required at the **MemConfig** pin is the inverse of this.

MemConfig is typically connected via an inverter to **MemnotWrD0**. Data bit zero of the least significant byte of each ROM word then provides the configuration data stream. By switching **MemConfig** between various data bus lines up to 32 configurations can be stored in ROM, one per bit of the data bus. **MemConfig** can be permanently connected to a data line or to **GND**. Connecting **MemConfig** to **GND** gives all **Tstates** configured to four periods; **notMemS1** pulse of maximum duration; **notMemS2-4** delayed by maximum; refresh interval 72 periods of **ClockIn**; refresh enabled; late write.

The external memory configuration table 6.10 shows the contribution of each memory address to the 13 configuration fields. The lowest 12 words (#7FFFFF6C to #7FFFFF98, fields 1 to 6) define the number of extra periods **Tm** to be added to each **Tstate**. If field 2 is 3 then three extra periods will be added to **T2** to extend it to the maximum of four periods.

The next five addresses (field 7) define the duration of **notMemS1** and the following fifteen (fields 8 to 10) define the delays before strobes **notMemS2-4** become active. The five bits allocated to each strobe allow durations of from 0 to 31 periods **Tm**, as described in strobes page 87.

Addresses #7FFFFFEC to #7FFFFFF4 (fields 11 and 12) define the refresh interval and whether refresh is to be used, whilst the final address (field 13) supplies a high bit to **MemConfig** if a late write cycle is required.

The columns to the right of the coding table show the values of each configuration bit for the four sample external configuration diagrams. Note the inclusion of period **E** at the end of **T6** in some diagrams. This is inserted to bring the start of the next **Tstate T1** to coincide with a rising edge of **ProcClockOut** (page 88).

Wait states **W** have been added to show the effect of them on strobe timing; they are not part of a configuration. In each case which includes wait states, two wait periods are defined. This shows that if a wait state would cause the start of **T5** to coincide with a falling edge of **ProcClockOut**, another period **Tm** is generated by the EMI to force it to coincide with a rising edge of **ProcClockOut**. This coincidence is only necessary if wait states are added, otherwise coincidence with a falling edge is permitted. Any configuration memory access is only permitted to be extended using wait, up to a total of 14 **ClockIn** periods.

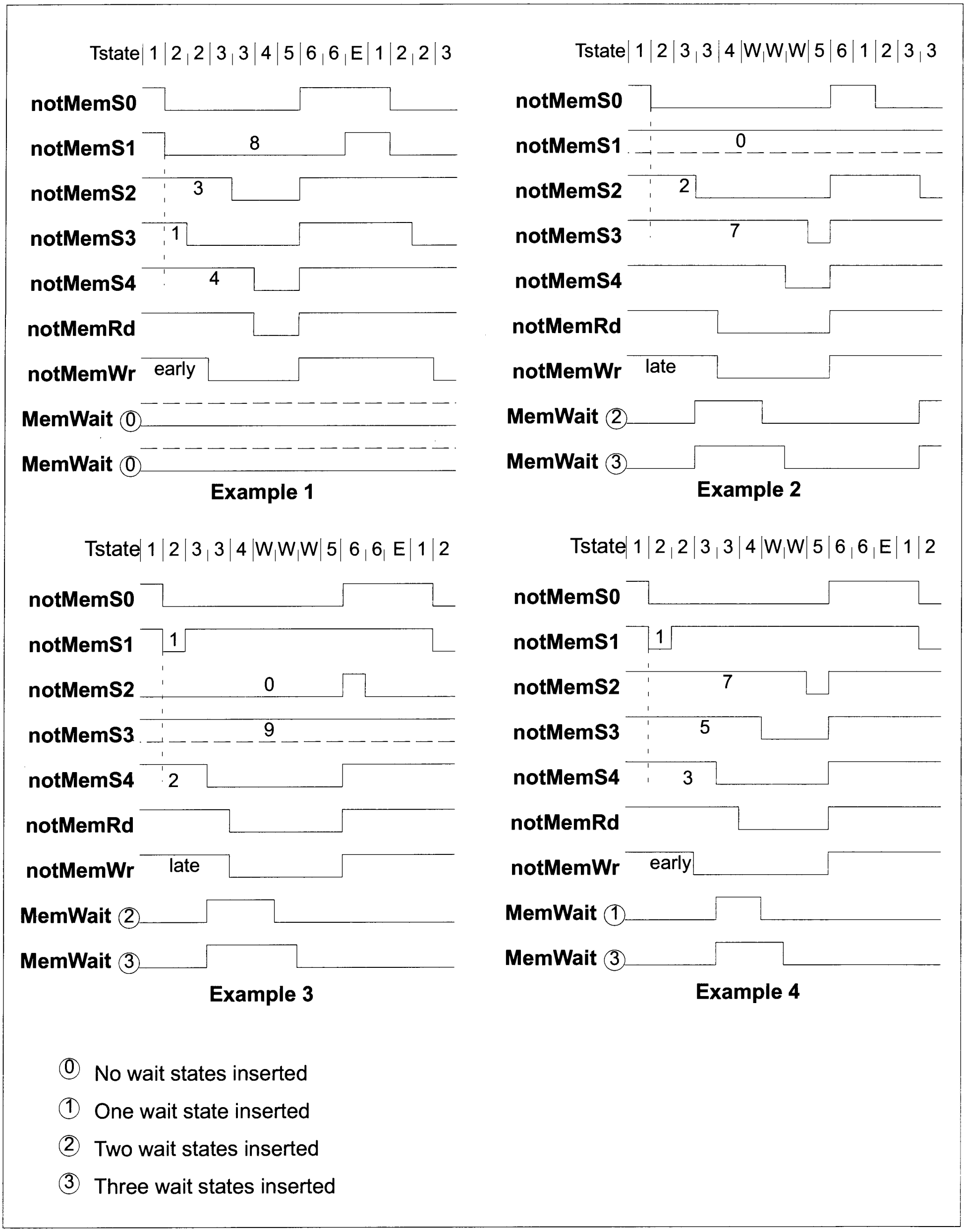
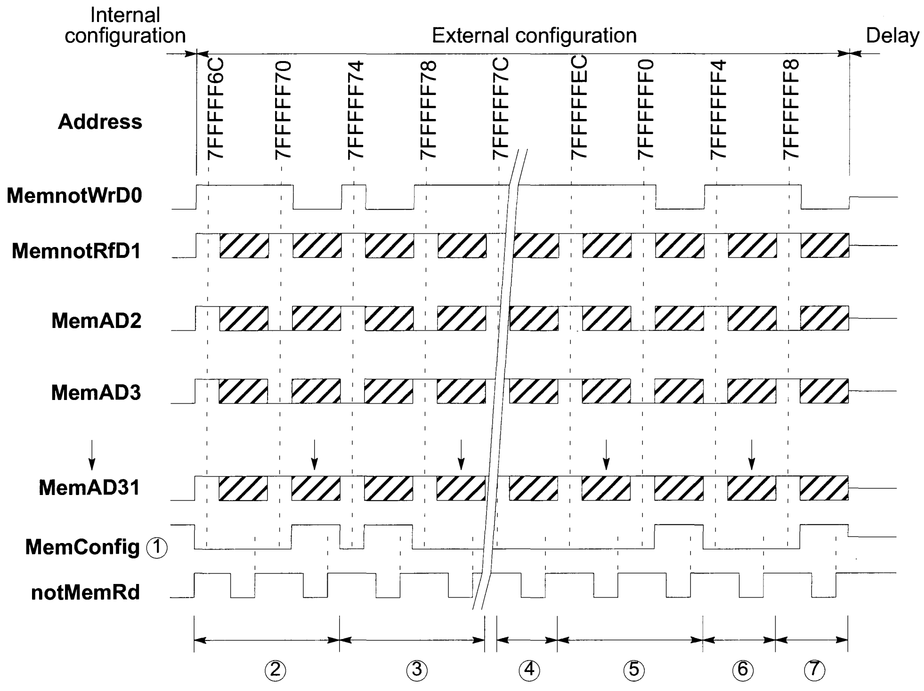


Figure 6.19 IMS T805 external configuration



① **MemConfig** connected to inverse of **MemnotWrD0**
② Configuration field 1; **T1** configured for 2 periods **Tm**
③ Configuration field 2; **T2** configured for 2 periods **Tm**
④ Configuration field 10; most significant bit of **notMemS4** configured high
⑤ Configuration field 11; refresh interval configured for 36 periods **ClockIn**
⑥ Configuration field 12; refresh enabled
⑦ Configuration field 13; early write cycle

Figure 6.20 IMS T805 external configuration scan

Scan	Mem AD	Field	Function	Example diagram			
cycle	address			1	2	3	4
1	7FFFFF6C	1	T1 least significant bit	0	0	0	0
2	7FFFFF70	1	T1 most significant bit	0	0	0	0
3	7FFFFF74	2	T2 least significant bit	1	0	0	1
4	7FFFFF78	2	T2 most significant bit	0	0	0	0
5	7FFFFF7C	3	T3 least significant bit	1	1	1	1
6	7FFFFF80	3	T3 most significant bit	0	0	0	0
7	7FFFFF84	4	T4 least significant bit	0	0	0	0
8	7FFFFF88	4	T4 most significant bit	0	0	0	0
9	7FFFFF8C	5	T5 least significant bit	0	0	0	0
10	7FFFFF90	5	T5 most significant bit	0	0	0	0
11	7FFFFF94	6	T6 least significant bit	1	0	1	1
12	7FFFFF98	6	T6 most significant bit	0	0	0	0
13	7FFFFF9C	7	notMemS1 least significant bit	0	0	1	1
14	7FFFFFA0	7		0	0	0	0
15	7FFFFFA4	7	⇓ ⇓	0	0	0	0
16	7FFFFFA8	7		1	0	0	0
17	7FFFFFAC	7	notMemS1 most significant bit	0	0	0	0
18	7FFFFFB0	8	notMemS2 least significant bit	1	0	0	1
19	7FFFFFB4	8		1	1	0	1
20	7FFFFFB8	8	⇓ ⇓	0	0	0	1
21	7FFFFFBC	8		0	0	0	0
22	7FFFFFC0	8	notMemS2 most significant bit	0	0	0	0
23	7FFFFFC4	9	notMemS3 least significant bit	1	1	1	1
24	7FFFFFC8	9		0	1	0	0
25	7FFFFFCC	9	⇓ ⇓	0	1	0	1
26	7FFFFFD0	9		0	0	1	0
27	7FFFFFD4	9	notMemS3 most significant bit	0	0	0	0
28	7FFFFFD8	10	notMemS4 least significant bit	0	0	0	1
29	7FFFFFDC	10		0	1	1	1
30	7FFFFFE0	10	⇓ ⇓	1	1	0	0
31	7FFFFFE4	10		0	0	0	0
32	7FFFFFE8	10	notMemS4 most significant bit	0	0	0	0
33	7FFFFFEC	11	Refresh Interval least significant bit	-	-	-	-
34	7FFFFFF0	11	Refresh Interval most significant bit	-	-	-	-
35	7FFFFFF4	12	Refresh Enable	-	-	-	-
36	7FFFFFF8	13	Late Write	0	1	1	0

Table 6.10 IMS T805 external configuration coding

Refresh interval	Interval in μs	Field 11 encoding	Complete cycle (ms)
18	3.6	00	0.922
36	7.2	01	1.843
54	10.8	10	2.765
72	14.4	11	3.686

Table 6.11 IMS T805 memory refresh configuration coding

Refresh intervals are in periods of **ClockIn** and **ClockIn** frequency is 5 MHz:

Interval = 18 * 200 = 3600 ns

Refresh interval is between successive incremental refresh addresses.
Complete cycles are shown for 256 row DRAMS.

Symbol	Parameter	T805-20		T805-25		T805-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TMCVRdH	**MemConfig** data setup	25		20		15		ns	
TRdHMCX	**MemConfig** data hold	0		0		0		ns	
TS0LRdH	**notMemS0** to configuration data read	388	412	308	332	257	277	ns	

Table 6.12 Memory configuration

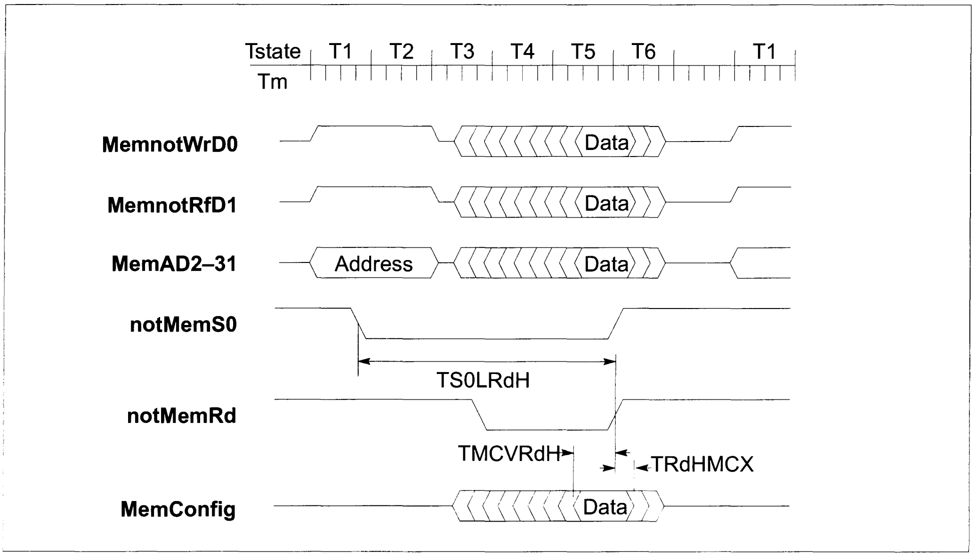


Figure 6.21 IMS T805 external configuration read cycle timing

7 Events

EventReq and **EventAck** provide an asynchronous handshake interface between an external event and an internal process. When an external event takes **EventReq** high the external event channel (additional to the external link channels) is made ready to communicate with a process. When both the event channel and the process are ready the processor takes **EventAck** high and the process, if waiting, is scheduled. **EventAck** is removed after **EventReq** goes low.

EventWaiting is asserted high by the transputer when a process executes an input on the event channel; typically with the occam `EVENT ? ANY` instruction. It remains high whilst the transputer is waiting for or servicing **EventReq** and is returned low when **EventAck** goes high. The **EventWaiting** pin changes near the falling edge of **ProcClockOut** and can therefore be sampled by the rising edge of **ProcClockOut**.

The **EventWaiting** pin can only be asserted by executing an *in* instruction on the event channel. The **EventWaiting** pin is not asserted high when an enable channel (*enbc*) instruction is executed on the Event channel (during an ALT construct in occam, for example). The **EventWaiting** pin can be asserted by executing the occam input on the event channel (such as `Event ? ANY`), provided that this does not occur as a guard in an alternative process. The **EventWaiting** pin can not be used to signify that an alternative process (ALT) is waiting on an input from the event channel.

EventWaiting allows a process to control external logic; for example, to clock a number of inputs into a memory mapped data latch so that the event request type can be determined.

Only one process may use the event channel at any given time. If no process requires an event to occur **EventAck** will never be taken high. Although **EventReq** triggers the channel on a transition from low to high, it must not be removed before **EventAck** is high. **EventReq** should be low during **Reset**; if not it will be ignored until it has gone low and returned high. **EventAck** is taken low when **Reset** occurs.

If the process is a high priority one and no other high priority process is running, the latency is as described on page 70. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

Cycle 1 Sample **EventReq** at pad on the rising edge of **ProcClockOut** and synchronise.

Cycle 2 Edge detect the synchronised **EventReq** and form the interrupt request.

Cycle 3 Sample interrupt vector for microcode ROM in the CPU.

Cycle 4 Execute the interrupt routine for Event rather than the next instruction.

Symbol	Parameter	T805-20		T805-25		T805-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TVHKH	**EventReq** response	0		0		0		ns	1
TKHVL	**EventReq** hold	0		0		0		ns	1
TVLKL	Delay before removal of **EventAck**	0	157	0	127	0	107	ns	
TKLVH	Delay before re-assertion of **EventReq**	0		0		0		ns	1
TKHEWL	**EventAck** to end of **EventWaiting**	0		0		0		ns	1

Notes

1 Guaranteed, but not tested.

Table 7.1 Event

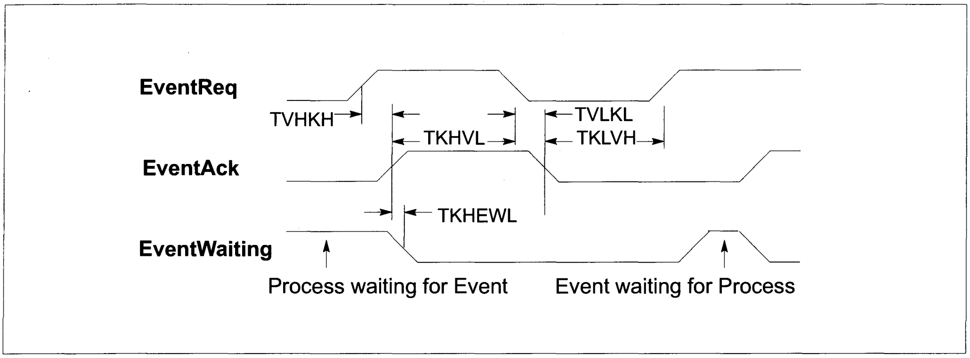


Figure 7.1 IMS T805 event timing

8 Links

Four identical INMOS bi-directional serial links provide synchronised communication between processors and with the outside world. Each link comprises an input channel and output channel. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

The IMS T805 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 5 or 20 Mbits/sec for 20 MHz and 25 MHz devices, and 20 Mbits/sec for faster devices. Links are not synchronised with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

Link speeds can be set by **LinkSpecial**, **Link0Special** and **Link123Special**. The link 0 speed can be set independently. Table 8.1 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed and as **Link123Special** for the others. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

Link Special	Linkn Special	Mbits/sec	Kbytes/sec	
			Uni	Bi
0	0	10	910	1250
0	1	5	450	670
1	0	10	910	1250
1	1	20	1740	2350

Table 8.1 Speed Settings for Transputer Links

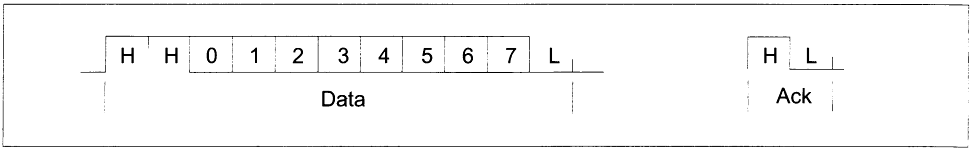


Figure 8.1 IMS T805 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	1
TJDf	**LinkIn** fall time				20	ns	1
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
		5 Mbits/s			30	ns	2
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		ohms	

Notes

1 Guaranteed, but not tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 8.2 Link

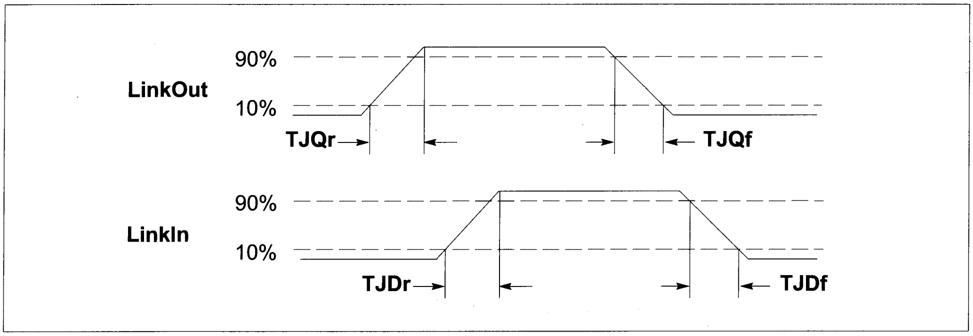


Figure 8.2 IMS T805 link timing

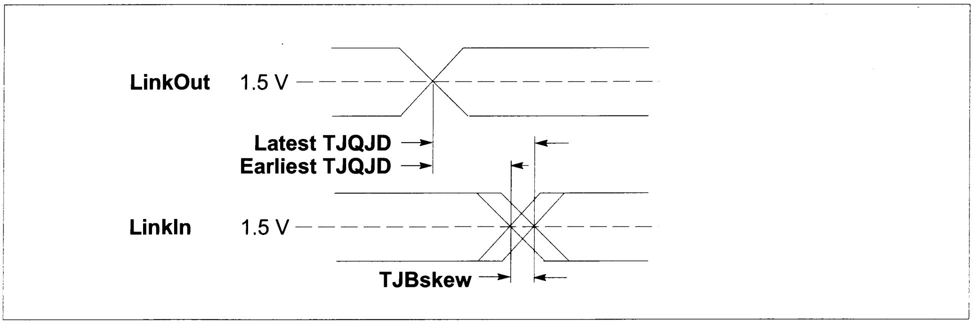


Figure 8.3 IMS T805 buffered link timing

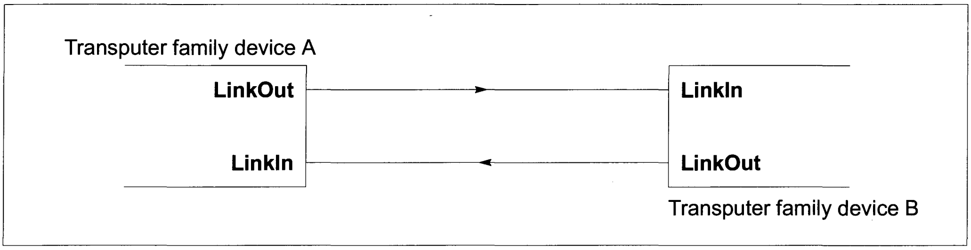


Figure 8.4 Links directly connected

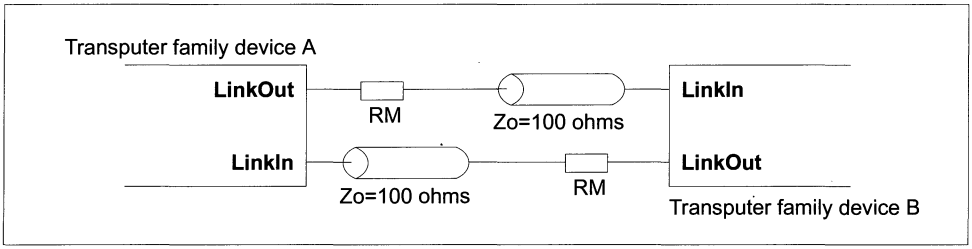


Figure 8.5 Links connected by transmission line

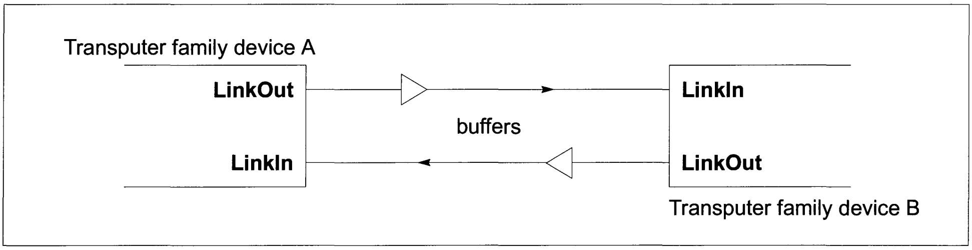


Figure 8.6 Links connected by buffers

9 Electrical specifications

9.1 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
II	Input current		±25	mA	4
tOSC	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 9.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1,2
CL	Load capacitance on any pin		60	pF	3
TA	Operating temperature range	0	70	°C	4

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Excluding **LinkOut** load capacitance.

4 Air flow rate 400 linear ft/min transverse air flow.

Table 9.2 Operating conditions

SYMBOL	PARAMETER		MIN	MAX	UNITS	NOTES
VIH	High level input voltage		2.0	VDD+0.5	V	1, 2
VIL	Low level input voltage		–0.5	0.8	V	1, 2
II	Input current	@ GND<VI<VDD		±10	μA	1, 2
VOH	Output high voltage	@ IOH=2mA	VDD–1		V	1, 2
VOL	Output low voltage	@ IOL=4mA		0.4	V	1, 2
IOZ	Tristate output current	@ GND<V0<VDD		±10	μA	1, 2
PD	Power dissipation			1.2	W	2, 3
CIN	Input capacitance	@ f=1MHz		7	pF	4
COZ	Output capacitance	@ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

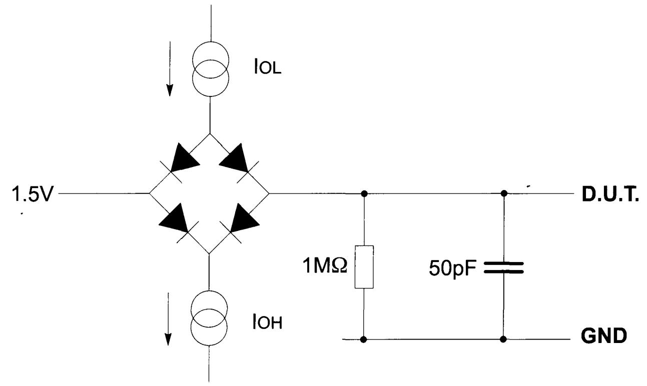
2 Parameters for IMS T805-S measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C. Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution. Power dissipation for processor operating at 20 MHz.

4 This parameter is sampled and not 100% tested.

Table 9.3 DC characteristics

9.2 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 9.1 Load circuit for AC measurements

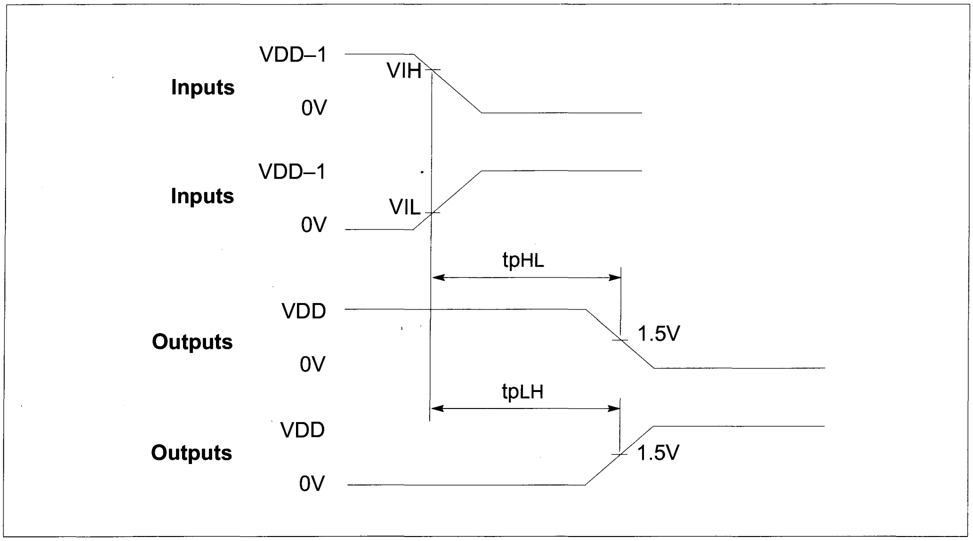


Figure 9.2 AC measurements timing waveforms

9.3 AC timing characteristics

Symbol	Parameter	Min	Max	Units	Notes
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3
TQr	Output rising edges		25	ns	1,3
TQf	Output falling edges		15	ns	1,3
TS0LaX	Address hold after **notMemS0**	**a**–8	**a**+8	ns	4

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 Guaranteed, but not tested.

4 **a** is **T2** where **T2** can be from one to four periods **Tm** in length.
Address lines include **MemnotWrD0**, **MemnotRfD1**, **MemAD2-31**.

Table 9.4 Input and output edges

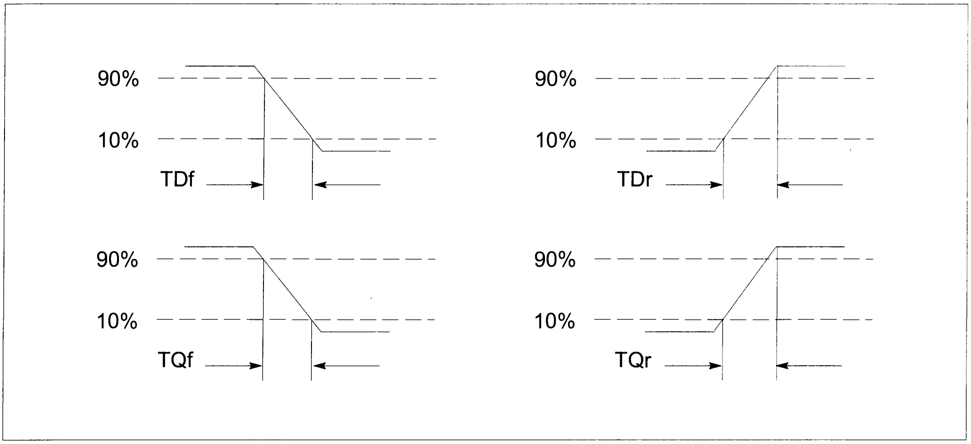


Figure 9.3 IMS T805 input and output edge timing

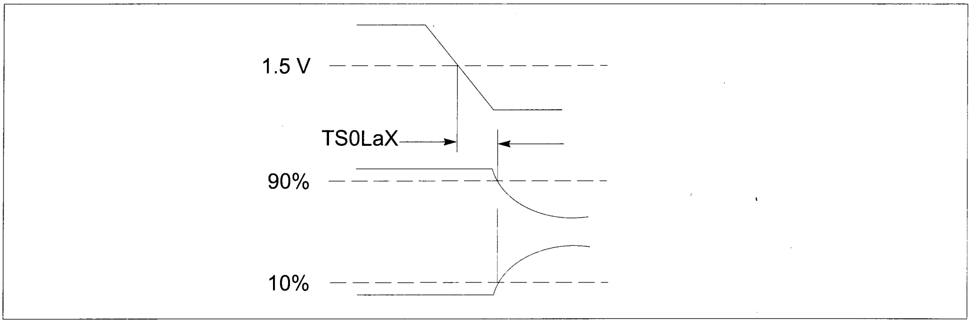
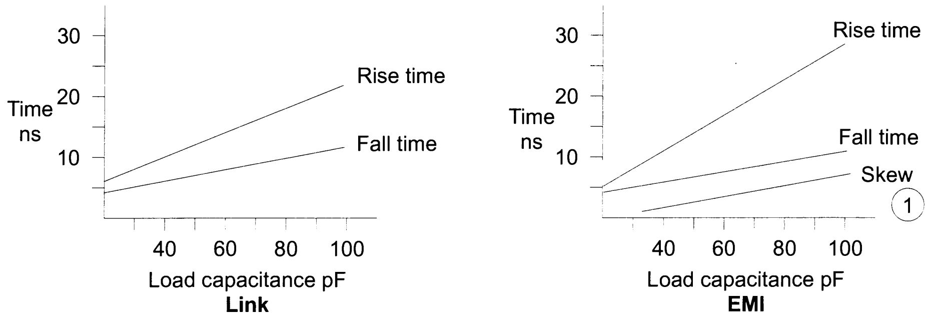


Figure 9.4 IMS T805 tristate timing relative to **notMemS0**



Notes

1 Skew is measured between **ProcClockOut** with a load of 2 Schottky TTL inputs and 30pF and **notMemS0** with a load of 2 Schottky TTL inputs and varying capacitance.

Figure 9.5 Typical rise/fall times

9.4 Power rating

Internal power dissipation (P_{INT}) of transputer and peripheral chips depends on **VDD**, as shown in figure 9.6. P_{INT} is substantially independent of temperature.

Total power dissipation (P_D) of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

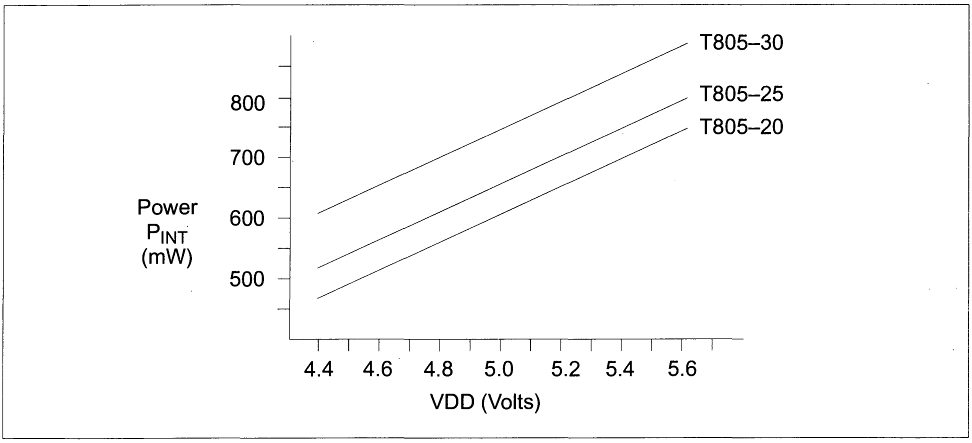
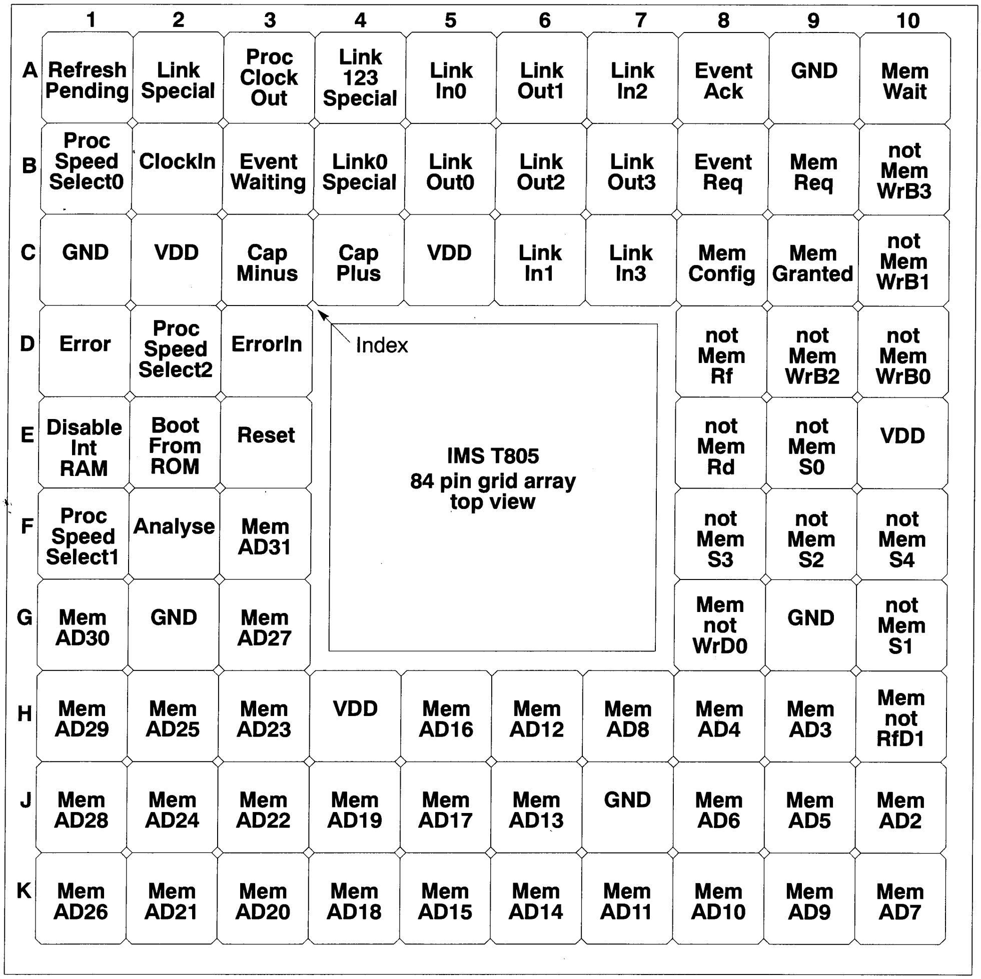


Figure 9.6 IMS T805 internal power dissipation vs VDD

10 Package pinouts

10.1 84 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Refresh Pending	Link Special	Proc Clock Out	Link 123 Special	Link In0	Link Out1	Link In2	Event Ack	GND	Mem Wait
B	Proc Speed Select0	ClockIn	Event Waiting	Link0 Special	Link Out0	Link Out2	Link Out3	Event Req	Mem Req	not Mem WrB3
C	GND	VDD	Cap Minus	Cap Plus	VDD	Link In1	Link In3	Mem Config	Mem Granted	not Mem WrB1
D	Error	Proc Speed Select2	ErrorIn					not Mem Rf	not Mem WrB2	not Mem WrB0
E	Disable Int RAM	Boot From ROM	Reset					not Mem Rd	not Mem S0	VDD
F	Proc Speed Select1	Analyse	Mem AD31					not Mem S3	not Mem S2	not Mem S4
G	Mem AD30	GND	Mem AD27					Mem not WrD0	GND	not Mem S1
H	Mem AD29	Mem AD25	Mem AD23	VDD	Mem AD16	Mem AD12	Mem AD8	Mem AD4	Mem AD3	Mem not RfD1
J	Mem AD28	Mem AD24	Mem AD22	Mem AD19	Mem AD17	Mem AD13	GND	Mem AD6	Mem AD5	Mem AD2
K	Mem AD26	Mem AD21	Mem AD20	Mem AD18	Mem AD15	Mem AD14	Mem AD11	Mem AD10	Mem AD9	Mem AD7

Figure 10.1 IMS T805 84 pin grid array package pinout

10.2 84 pin PLCC J-bend package

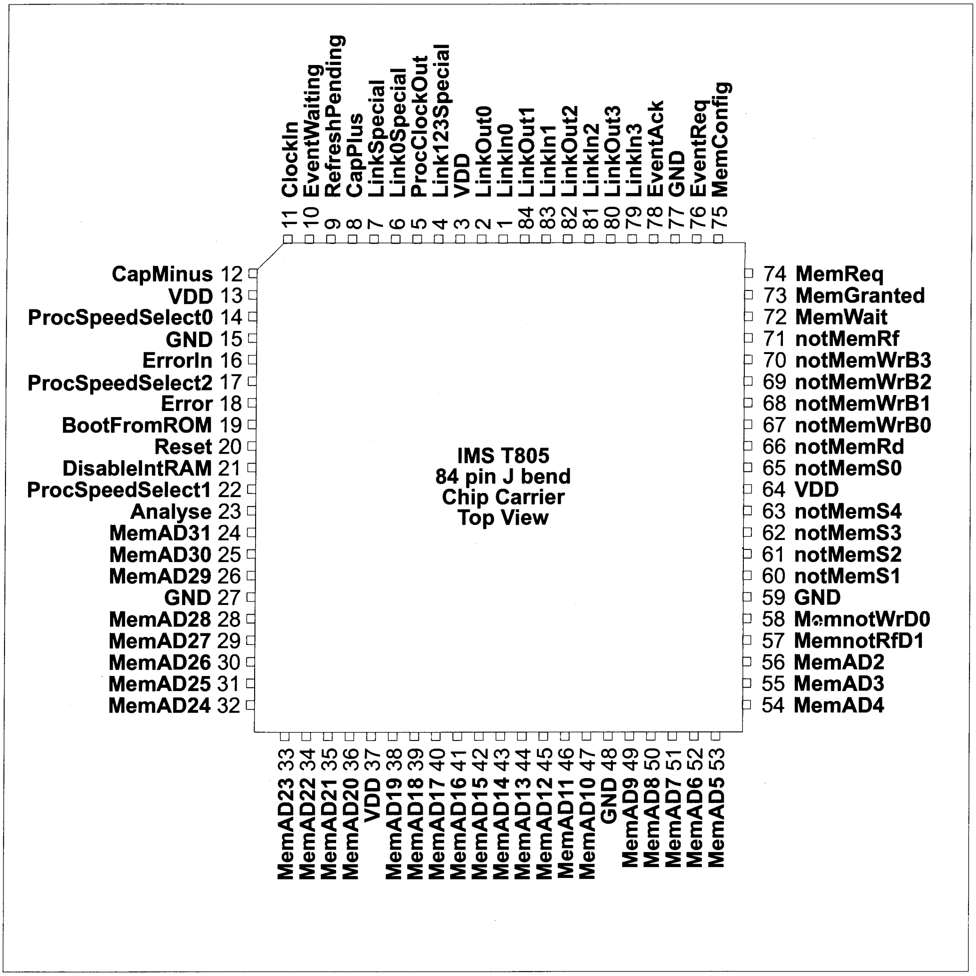
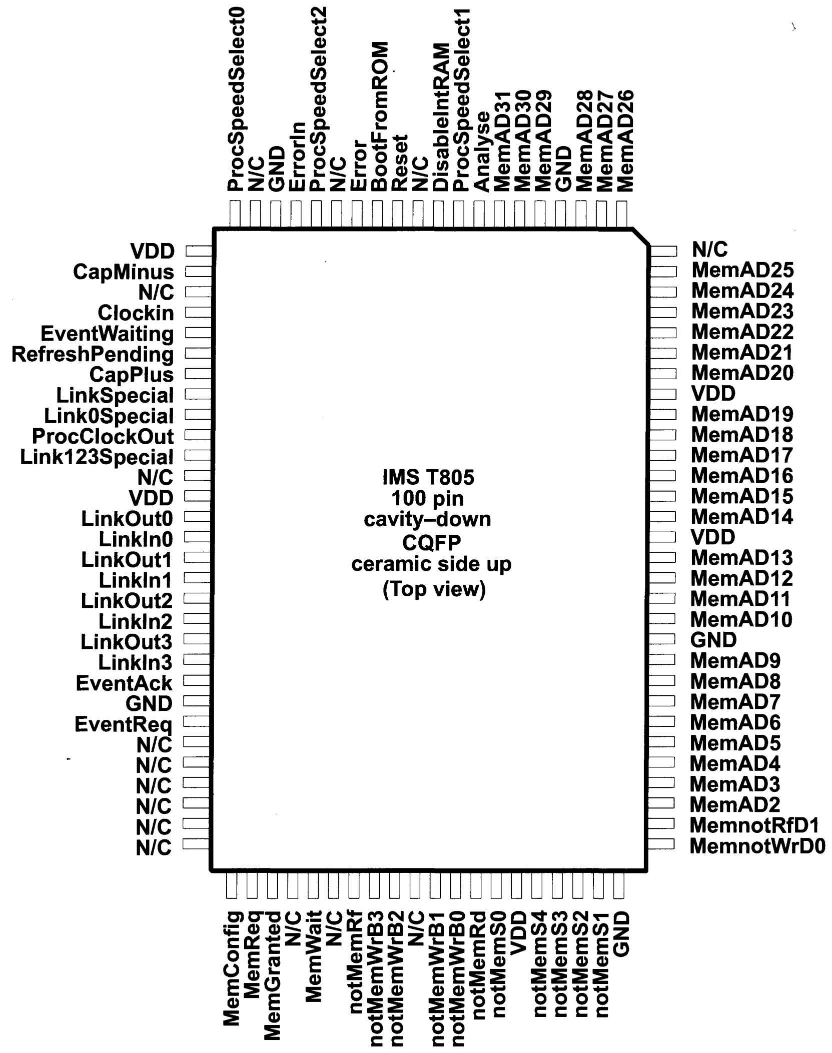


Figure 10.2 IMS T805 84 pin PLCC J-bend package pinout

10.3 100 pin cavity-down ceramic quad flat pack package



N/C indicates pins not connected

Figure 10.3 IMS T805 100 pin cavity-down ceramic quad flat pack package pinout

11 Ordering

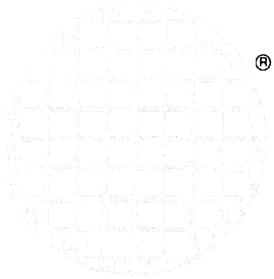
This section indicates the designation of speed and package selections for the various devices. Speed of **ClockIn** is 5 MHz for all parts. Transputer processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in the external memory section.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	**Processor clock speed**	**Processor cycle time**	**PLLx**	**Package**
IMS T805-G20S	20.0	50	4.0	84 pin ceramic pin grid array
IMS T805-G25S	25.0	40	5.0	84 pin ceramic pin grid array
IMS T805-G30S	30.0	33	6.0	84 pin ceramic pin grid array
IMS T805-J20S	20.0	50	4.0	84 pin PLCC J-bend
IMS T805-J25S	25.0	40	5.0	84 pin PLCC J-bend
IMS T805-F20S	20.0	50	4.0	100 pin ceramic quad flat pack
IMS T805-F25S	25.0	40	5.0	100 pin ceramic quad flat pack
IMS T805-F30S	30.0	33	6.0	100 pin ceramic quad flat pack

Table 11.1 IMS T805 ordering details

Military versions to MIL-STD-883 are available, see *'The Military and Space Transputer Databook'* for details.



IMS T801 transputer

inmos®

Preliminary Data

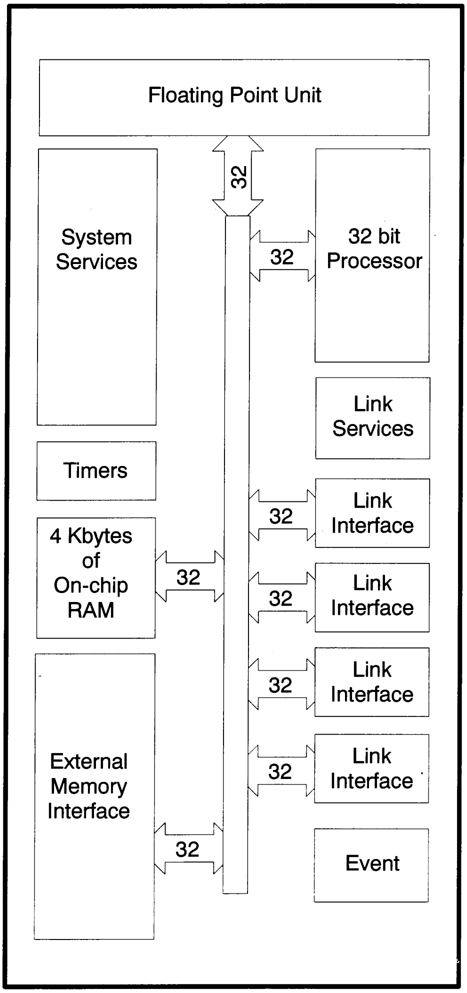
The information in this datasheet is subject to change

FEATURES

32 bit architecture
40 ns internal cycle time
25 MIPS (peak) instruction rate
4.3 Mflops (peak) instruction rate
Debugging support
64 bit on-chip floating point unit which conforms to IEEE 754
4 Kbytes on-chip static RAM
120 Mbytes/sec sustained data rate to internal memory
4 Gbytes directly addressable external memory
50 Mbytes/sec sustained data rate to external memory
750 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
Bi-directional data rate of 2.4 Mbytes/sec per link
High performance graphics support with block move instructions
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ±5% power supply
Packaging 100 pin PGA

APPLICATIONS

Scientific and mathematical applications
High speed multi processor systems
High performance graphics processing
Supercomputers
Workstations and workstation clusters
Digital signal processing
Accelerator processors
Distributed databases
System simulation
Telecommunications
Robotics
Fault tolerant systems
Image processing
Pattern recognition
Artificial intelligence



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992
42 1441 01

1 Introduction

The IMS T801 transputer is a 32 bit CMOS microcomputer with a 64 bit floating point unit and graphics support. It has 4 Kbytes on-chip RAM for high speed processing, a 32 bit non-multiplexed external memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages and provides direct support for the occam model of concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond.

For convenience of description, the IMS T801 operation is split into the basic blocks shown in figure 1.1.

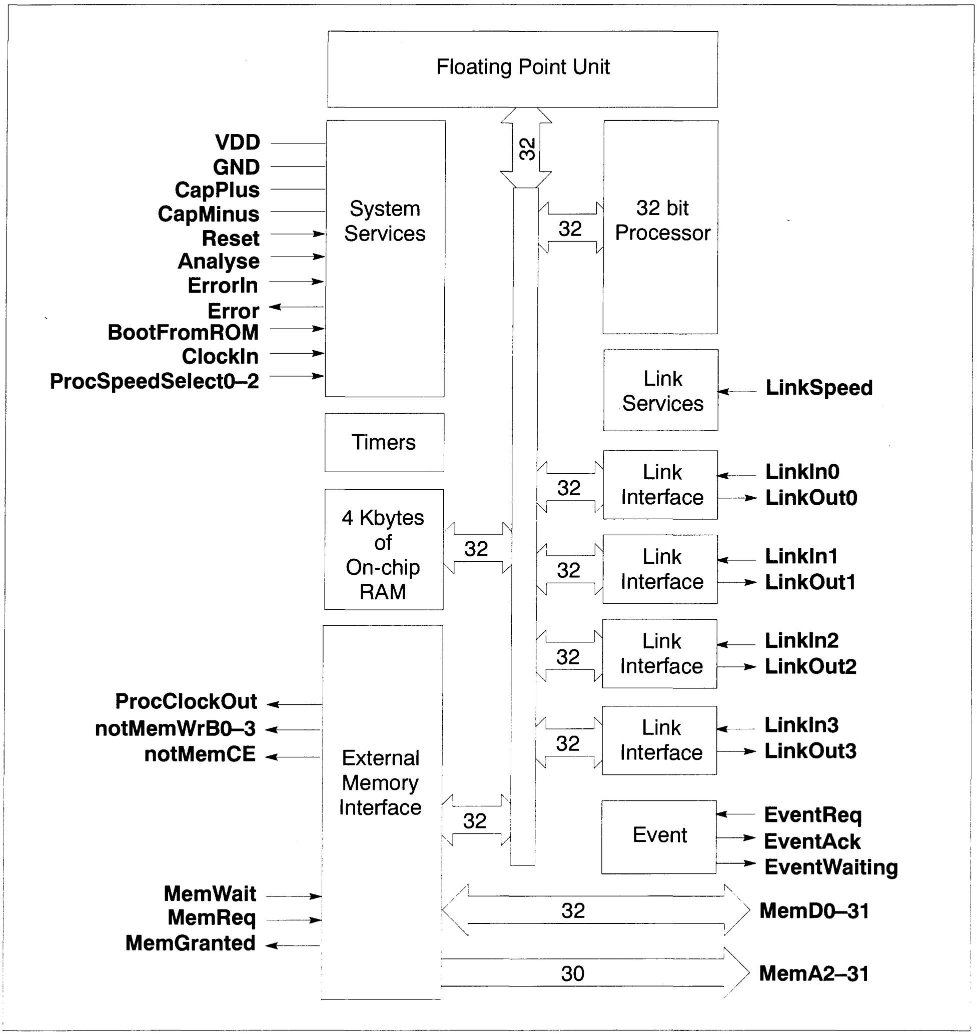


Figure 1.1 IMS T801 block diagram

The processor speed of a device can be pin-selected in stages from 20 MHz up to the maximum allowed for the part. A device running at 25 MHz achieves an instruction throughput of 25 MIPS peak and 17.5 MIPS sustained.

The IMS T801 provides high performance arithmetic and floating point operations. The 64 bit floating point unit provides single and double length operation to the ANSI-IEEE 754-1985 standard for floating point arithmetic. It is able to perform floating point operations concurrently with the processor, sustaining a rate of 2.2 Mflops at a processor speed of 20 MHz.

High performance graphics support is provided by microcoded block move instructions which operate at the speed of memory. The two-dimensional block move instructions provide for contiguous block moves as well as block copying of either non-zero bytes of data only or zero bytes only. Block move instructions can be used to provide graphics operations such as text manipulation, windowing, panning, scrolling and screen updating.

Cyclic redundancy checking (CRC) instructions are available for use on arbitrary length serial data streams, to provide error detection where data integrity is critical. Another feature of the IMS T801, useful for pattern recognition, is the facility to count bits set in a word.

The IMS T801 can directly access a linear address space of 4 Gbytes. The 32 bit wide memory interface uses non-multiplexed data and address lines and provides a data rate of up to 4 bytes every 100 nanoseconds (50 Mbytes/sec) for a 25 MHz device.

System Services include processor reset and bootstrap control, together with facilities for error analysis.

The standard INMOS communication links allow networks of transputer family products to be constructed by direct point to point connections with no external logic. The IMS T801 links support the standard operating speed of 10 Mbits/sec, but also operate at 20 Mbits/sec. Each link can transfer data bi-directionally at up to 2.35 Mbytes/sec. The transputer is designed to implement the occam language, detailed in the occam Reference Manual, but also efficiently supports other languages such as C, Pascal and Fortran. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*, where the IMS T800 instruction set is applicable. This data sheet supplies hardware implementation and characterisation details for the IMS T801. It is intended to be read in conjunction with the Transputer Architecture chapter, which details the architecture of the transputer and gives an overview of occam.

The IMS T801 instruction set contains a number of instructions to facilitate the implementation of breakpoints. For further information concerning breakpointing, refer to *Support for debugging/breakpointing in transputers* (technical note 61).

Figure 1.2 shows the internal datapaths for the IMS T801.

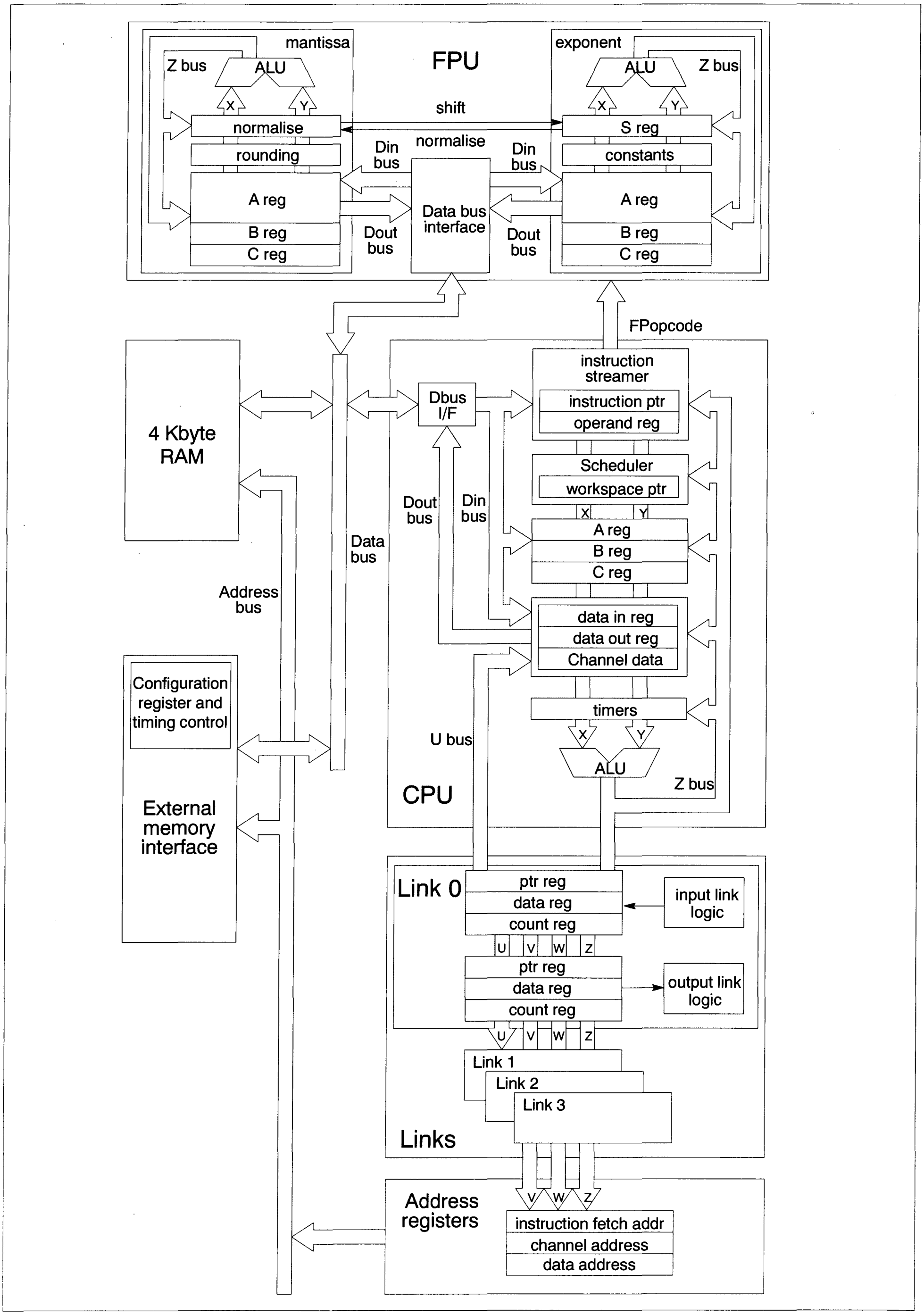


Figure 1.2 IMS T801 internal datapaths

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high.
Pinout details for various packages are given on page 161.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
ProcSpeedSelect0-2	in	Processor speed selectors
Reset	in	System reset
ErrorOut	out	Error indicator
Analyse	in	Error analysis
BootFromROM	in	Boot from external ROM or from link

Table 2.1 IMS T801 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemA2-31	out	Thirty address lines
Data0-31	in/out	Thirty-two non-multiplexed data lines
notMemWrB0-3	out	Four byte-addressing write strobes
notMemCE	out	Chip enable
MemWait	in	Memory cycle extender
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted

Table 2.2 IMS T801 external memory interface

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge
EventWaiting	out	Event input requested by software

Table 2.3 IMS T801 event

Pin	In/Out	Function
LinkIn0-3	in	Four serial data input channels
LinkOut0-3	out	Four serial data output channels
LinkSpeed	in	Select speed for Links 0-3 to 10 or 20 Mbits/sec

Table 2.4 IMS T801 link

3 Floating point unit

The 64 bit FPU provides single and double length arithmetic to floating point standard ANSI-IEEE 754-1985. It is able to perform floating point arithmetic concurrently with the central processor unit (CPU), sustaining 1.8 Mflops on a 25 MHz device. All data communication between memory and the FPU occurs under control of the CPU.

The FPU consists of a microcoded computing engine with a three deep floating point evaluation stack for manipulation of floating point numbers. These stack registers are **FA**, **FB** and **FC**, each of which can hold either 32 bit or 64 bit data; an associated flag, set when a floating point value is loaded, indicates which. The stack behaves in a similar manner to the CPU stack (page 26).

As with the CPU stack, the FPU stack is not saved when rescheduling occurs. The FPU can be used in both low and high priority processes. When a high priority process interrupts a low priority one the FPU state is saved inside the FPU. The CPU will service the interrupt immediately on completing its current operation. The high priority process will not start, however, before the FPU has completed its current operation.

Points in an instruction stream where data need to be transferred to or from the FPU are called *synchronisation points*. At a synchronisation point the first processing unit to become ready will wait until the other is ready. The data transfer will then occur and both processors will proceed concurrently again. In order to make full use of concurrency, floating point data source and destination addresses can be calculated by the CPU whilst the FPU is performing operations on a previous set of data. Device performance is thus optimised by minimising the CPU and FPU idle times.

The FPU has been designed to operate on both single length (32 bit) and double length (64 bit) floating point numbers, and returns results which fully conform to the ANSI-IEEE 754-1985 floating point arithmetic standard. Denormalised numbers are fully supported in the hardware. All rounding modes defined by the standard are implemented, with the default being round to nearest.

The basic addition, subtraction, multiplication and division operations are performed by single instructions. However, certain less frequently used floating point instructions are selected by a value in register *A* (when allocating registers, this should be taken into account). A *load constant* instruction *ldc* is used to load register *A*; the *floating point entry* instruction *fpentry* then uses this value to select the floating point operation. This pair of instructions is termed a *selector sequence*.

Names of operations which use *fpentry* begin with *fpu*. A typical usage, returning the absolute value of a floating point number, would be

fpuabs; *ldc* *fpentry;*

Since the indirection code for *fpuabs* is **0B**, it would be encoded as

Mnemonic		**Function code**	**Memory code**
ldc	*fpuabs*	#4	#4B
fpentry	(op. code #AB)		#2AFB
is coded as			
pfix	#A	#2	#2A
opr	#B	#F	#FB

The *remainder* and *square root* instructions take considerably longer than other instructions to complete. In order to minimise the interrupt latency period of the transputer they are split up to form instruction sequences. As an example, the instruction sequence for a single length square root is

fpusqrtfirst; *fpusqrtstep;* *fpusqrtstep;* *fpusqrtlast;*

The FPU has its own error flag *FP_Error*. This reflects the state of evaluation within the FPU and is set in circumstances where invalid operations, division by zero or overflow exceptions to the ANSI-IEEE 754-1985 standard would be flagged (page 49). *FP_Error* is also set if an input to a floating point operation is infinite or is not a number (NaN). The *FP_Error* flag can be set, tested and cleared without affecting the main *Error* flag, but can also set *Error* when required (page 48). Depending on how a program is compiled, it is possible for both unchecked and fully checked floating point arithmetic to be performed.

Further details on the operation of the FPU can be found in *Transputer Instruction Set–A Compiler Writer's Guide*.

	T801-20		T801-25	
Operation	**Single length**	**Double length**	**Single length**	**Double length**
add	350 ns	350 ns	292 ns	292 ns
subtract	350 ns	350 ns	292 ns	292 ns
multiply	550 ns	1000 ns	459 ns	834 ns
divide	850 ns	1600 ns	709 ns	1333 ns
Timing is for operations where both operands are normalised fp numbers.				

Table 3.1 Typical floating point operation times for IMS T801

4 System services

System services include all the necessary logic to initialise and sustain operation of the device. They also include error handling and analysis facilities.

4.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

4.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1 μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 10 MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

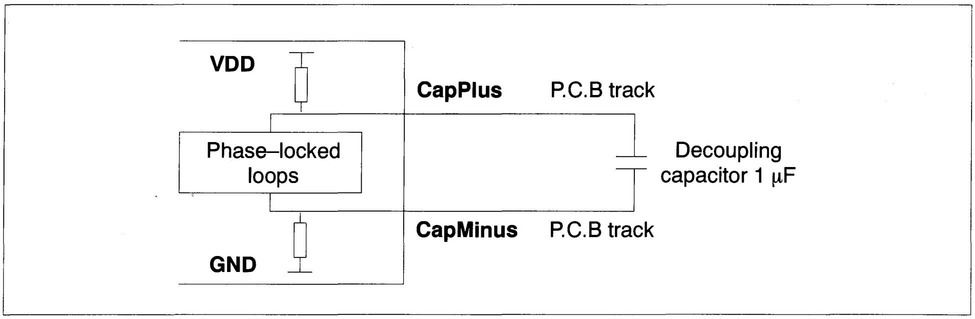


Figure 4.1 Recommended PLL decoupling

4.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	T801-20, -25			Units	Notes
		Min	Nom	Max		
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1, 2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1, 3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1, 4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200 ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 9.3).

Table 4.1 Input clock

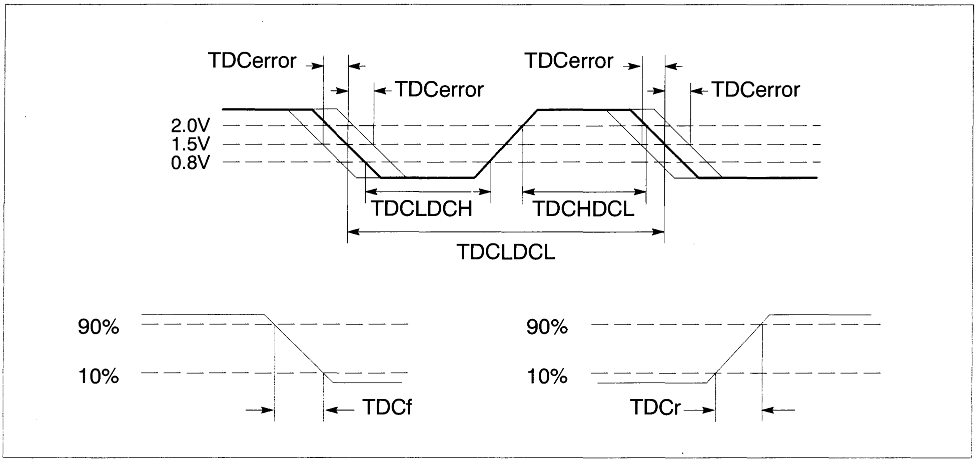


Figure 4.2 ClockIn timing

4.4 ProcSpeedSelect0–2

Processor speed of the IMS T801 is variable in discrete steps. The desired speed can be selected, up to the maximum rated for a particular component, by the three speed select lines **ProcSpeedSelect0-2**. The pins are tied high or low, according to table 4.2, for the various speeds. The frequency of **ClockIn** for the speeds given in table 4.2 is 5 MHz. There are six valid speed select combinations.

ProcSpeed-Select2	ProcSpeed-Select1	ProcSpeed-Select0	Processor Clock Speed MHz	Processor Cycle Time ns	Notes
0	0	0	20.0	50.0	
0	0	1	22.5	44.4	Not supported
0	1	0	25.0	40.0	
0	1	1	30.0	33.3	Not supported
1	0	0	35.0	28.6	Not supported
1	0	1			Invalid
1	1	0	17.5	57.1	Not supported
1	1	1			Invalid

Table 4.2 Processor speed selection

4.5 Bootstrap

The transputer can be bootstrapped either from a link or from external ROM. To facilitate debugging, **BootFromROM** may be dynamically changed but must obey the specified timing restrictions. It is sampled once only by the transputer, before the first instruction is executed after **Reset** is taken low.

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFFFFFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the *W* register points to *MemStart* (page 140).

If **BootFromROM** is connected low (e.g. to **GND**) the transputer will wait for the first bootstrap message to arrive on any one of its links. The transputer is ready to receive the first byte on a link within two processor cycles **TPCLPCL** after **Reset** goes low.

If the first byte received (the control byte) is greater than 1 it is taken as the quantity of bytes to be input. The following bytes, to that quantity, are then placed in internal memory starting at location *MemStart*. Following reception of the last byte the transputer will start executing code at *MemStart* as a low priority process. **BootFromROM** may be taken high after reception of the last byte, if required. The memory space immediately above the loaded code is used as work space. A byte arriving on other links after the control byte has been received and on the bootstrapping link after the last bootstrap byte, will be retained and no acknowledge will be sent until a process inputs from them.

4.6 Peek and poke

Any location in internal or external memory can be interrogated and altered when the transputer is waiting for a bootstrap from link. If the control byte is 0 then eight more bytes are expected on the same link. The first four byte word is taken as an internal or external memory address at which to poke (write) the second four byte word. If the control byte is 1 the next four bytes are used as the address from which to peek (read) a word of data; the word is sent down the output channel of the same link.

Following such a peek or poke, the transputer returns to its previously held state. Any number of accesses may be made in this way until the control byte is greater than 1, when the transputer will commence reading its bootstrap program. Any link can be used, but addresses and data must be transmitted via the same link as the control byte.

4.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initialises the transputer and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) is ignored whilst **Reset** is high but can occur before bootstrap (page 149).

If **BootFromROM** is high bootstrapping will then take place immediately, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

4.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 48). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags *Error*, *HaltOnError* and *EnableJ0Break* are normally cleared at reset on the IMS T801; however, if **Analyse** is asserted the flags are not altered. Memory refresh continues.

Input links will continue with outstanding transfers. Output links will not make another access to memory for data but will transmit only those bytes already in the link buffer. Providing there is no delay in link acknowledgement, the links should be inactive within a few microseconds of the transputer halting.

Reset should not be asserted before the transputer has halted and link transfers have ceased. When **Reset** is taken low whilst **Analyse** is high, neither the memory configuration sequence nor the block of eight refresh cycles will occur; the previous memory configuration will be used for any external memory accesses. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 4.3.

I	**MemStart** if bootstrapping from a link, or the external memory bootstrap address if bootstrapping from ROM.
W	**MemStart** if bootstrapping from ROM, or the address of the first free word after the bootstrap program if bootstrapping from link.
A	The value of **I** when the processor halted.
B	The value of **W** when the processor halted, together with the priority of the process when the transputer was halted (i.e. the **W** descriptor).
C	The ID of the bootstrapping link if bootstrapping from link.

Table 4.3 Register values after Analyse

Symbol	Parameter	T801-20, -25			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

1 Full periods of **ClockIn TDCLDCL** required.

2 At power-on reset.

3 Must be stable until after end of bootstrap period. See Bootstrap section 4.5.

Table 4.4 **Reset** , **Analyse** and **BootFromROM** timing

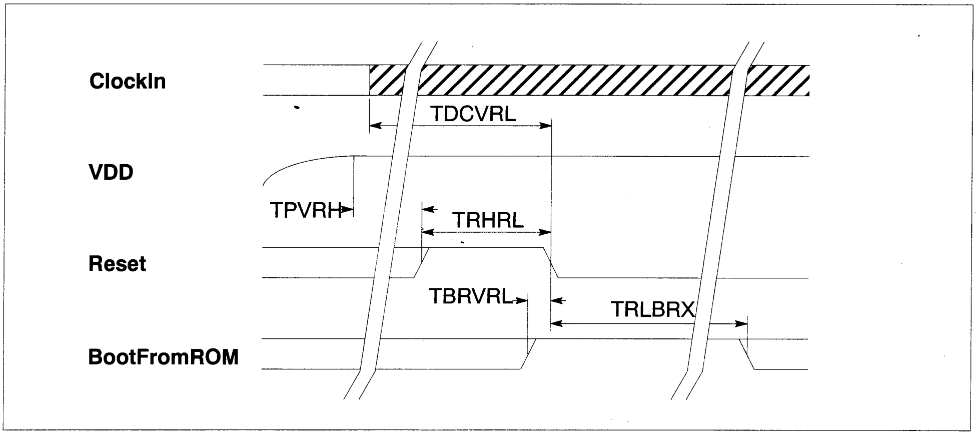


Figure 4.3 Transputer **Reset** timing with **Analyse** low

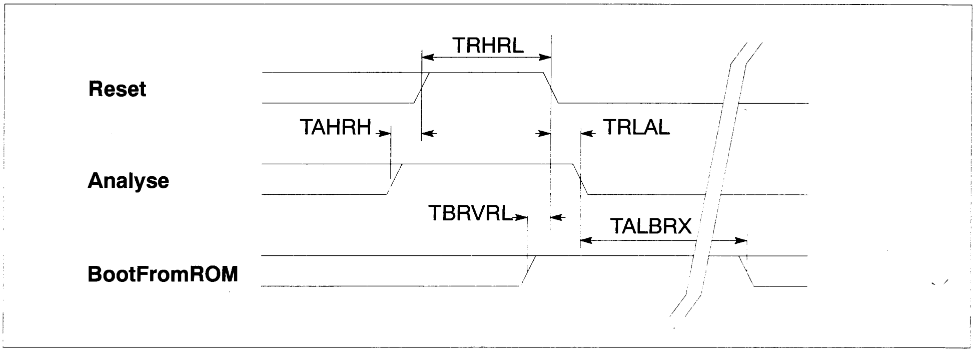


Figure 4.4 Transputer **Reset**, **Analyse** and **BootFromROM** timing

4.9 ErrorOut

The **ErrorOut** pin is connected directly to the internal *Error* flag and follows the state of that flag. If **ErrorOut** is high it indicates an error in one of the processes caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 48). It can also be set from the floating point unit under certain circumstances (page 49, 132). Once set, the *Error* flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (page 137).

A process can be programmed to stop if the *Error* flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data.

By setting the *HaltOnError* flag the transputer itself can be programmed to halt if *Error* becomes set. If *Error* becomes set after *HaltOnError* has been set, all processes on that transputer will cease but will not necessarily cause other transputers in a network to halt. Setting *HaltOnError* after *Error* will not cause the transputer to halt; this allows the processor reset and analyse facilities to function with the flags in indeterminate states.

An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by applying the **ErrorOut** output signal of the errant transputer to the **EventReq** pin of a suitably programmed master transputer. Since the process state is preserved when stopped by an error, the master transputer can then use the analyse function to debug the fault. When using such a circuit, note that the *Error* flag is in an indeterminate state on power up; the circuit and software should be designed with this in mind.

Error checks can be removed completely to optimise the performance of a proven program; any unexpected error then occurring will have an arbitrary undefined effect.

If a high priority process pre-empts a low priority one, status of the *Error* and *HaltOnError* flags is saved for the duration of the high priority process and restored at the conclusion of it. Status of the *Error* flag is transmitted to the high priority process but the *HaltOnError* flag is cleared before the process starts. Either flag can be altered in the process without upsetting the error status of any complex operation being carried out by the pre-empted low priority process.

In the event of a transputer halting because of *HaltOnError*, the links will finish outstanding transfers before shutting down. If **Analyse** is asserted then all inputs continue but outputs will not make another access to memory for data. Memory refresh will continue to take place.

After halting due to the *Error* flag changing from 0 to 1 whilst *HaltOnError* is set, register *I* points two bytes past the instruction which set *Error*. After halting due to the **Analyse** pin being taken high, register *I* points one byte past the instruction being executed. In both cases *I* will be copied to register *A*.

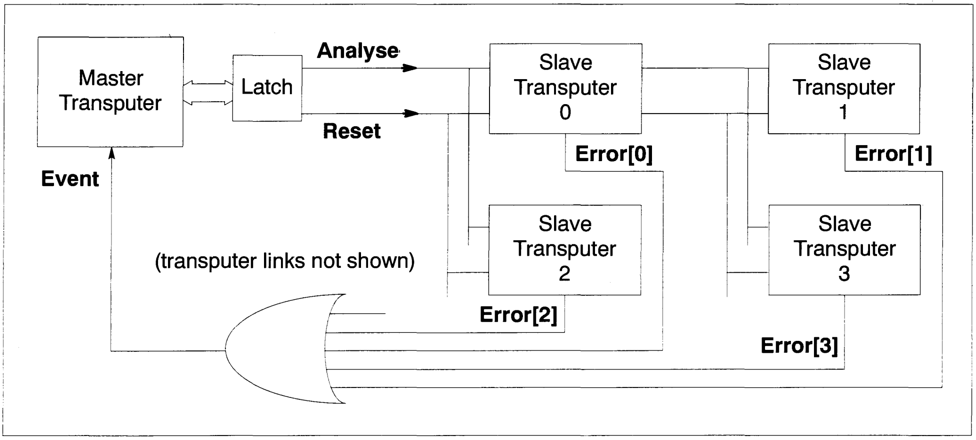


Figure 4.5 Error handling in a multi–transputer system

5 Memory

The IMS T801 can access 4 Gbytes of external memory space. The IMS T801 also has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (page 144). Internal and external memory are part of the same linear address space.

IMS T801 memory is byte addressed, with words aligned on four-byte boundaries. The least significant byte of a word is the lowest addressed byte.

The bits in a byte are numbered 0 to 7, with bit 0 the least significant. The bytes are numbered from 0, with byte 0 the least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

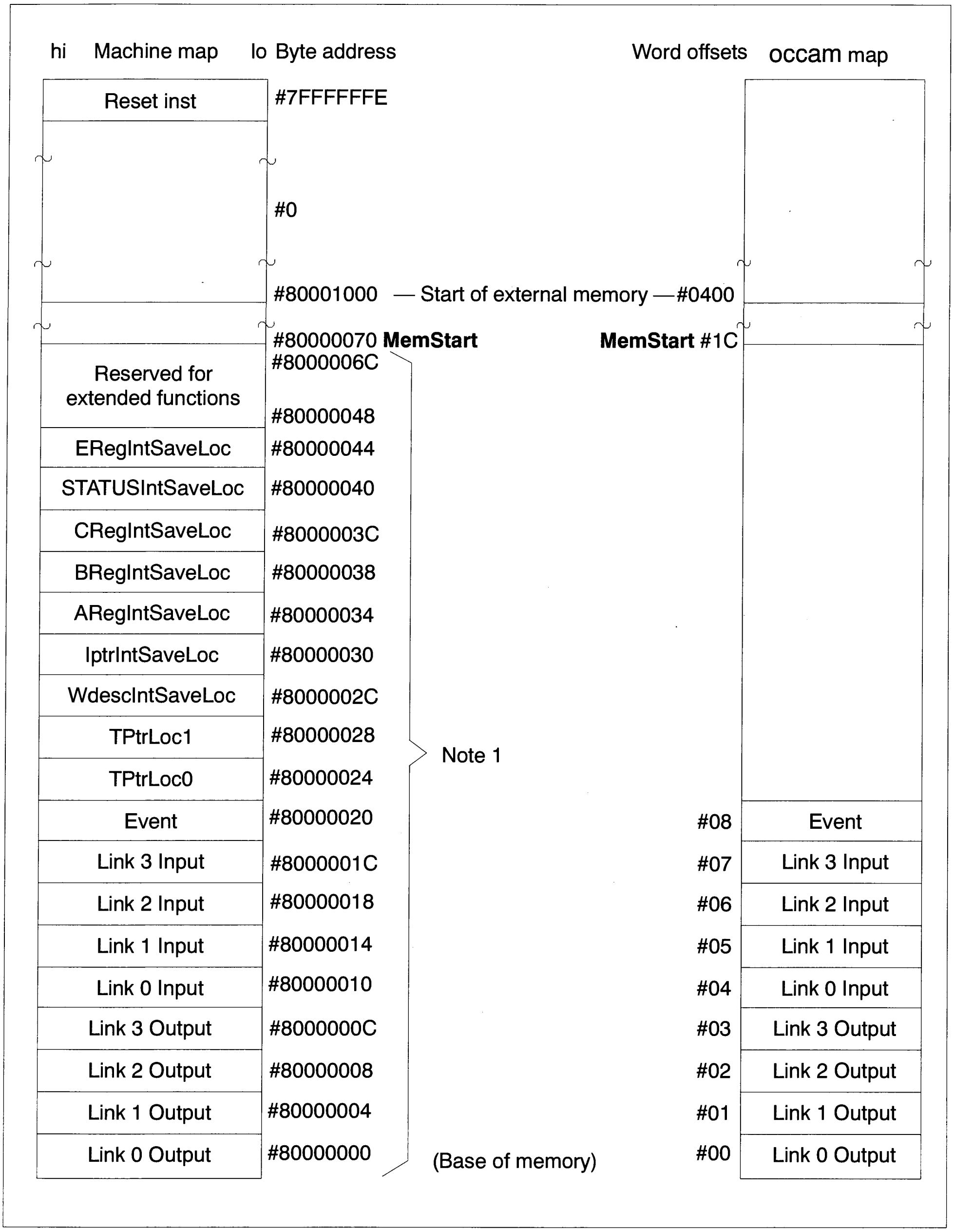
Internal memory starts at the most negative address #80000000 and extends to #80000FFF. User memory begins at #80000070; this location is given the name *MemStart*.

The reserved area of internal memory below *MemStart* is used to implement link and event channels.

Two words of memory are reserved for timer use, *TPtrLoc0* for high priority processes and *TPtrLoc1* for low priority processes. They either indicate the relevant priority timer is not in use or point to the first process on the timer queue at that priority level.

Values of certain processor registers for the current low priority process are saved in the reserved *IntSaveLoc* locations when a high priority process pre-empts a low priority one. Other locations are reserved for extended features such as block moves and floating point operations.

External memory space starts at #80001000 and extends up through #00000000 to #7FFFFFFF. ROM bootstrapping code must be in the most positive address space, starting at #7FFFFFFE. Address space immediately below this is conventionally used for ROM based code.



Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 137). For details see *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 5.1 IMS T801 memory map

6 External memory interface

The IMS T801 External Memory Interface (EMI) allows access to a 32 bit address space via separate address and data buses.

The external memory cycle is divided into four **Tstates** with the following functions:

T1 Address and control setup time.

T2 Data setup time.

T3 Data read/write.

T4 Data and address hold after access.

Each **Tstate** is half a processor cycle **TPCLPCL** long. An external memory cycle is always a complete number of cycles **TPCLPCL** in length. The start of **T1** always coincides with a rising edge of **ProcClockOut**. **T2** can be extended indefinitely by adding externally generated wait states of one complete processor cycle each.

During an internal memory access cycle the external memory interface address bus **MemA2-31** reflects the word address used to access internal RAM, **notMemWrB0-3** and **notMemCE** are inactive and the data bus **MemD0-31** is tristated. This is true unless and until a DMA (memory request) activity takes place, when the **MemA2-31**, **MemD0-31**, **notMemCE** and **notMemWrB0-3** signals will be placed in a high impedance state by the transputer.

Bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 136).

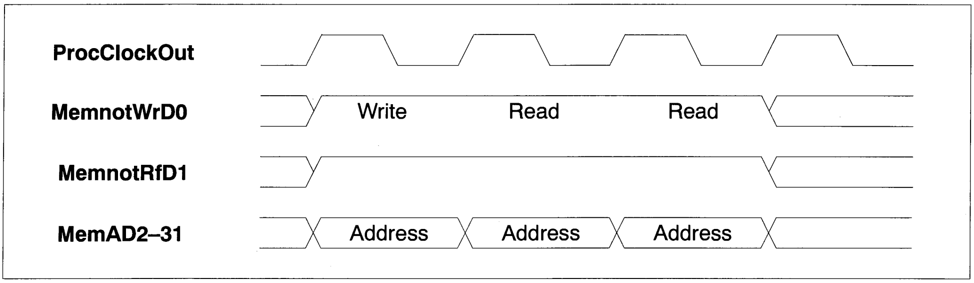


Table 6.1 IMS T801 bus activity for internal memory cycles

6.1 Pin functions

6.1.1 MemA2–31

External memory addresses are output on a non-multiplexed 30 bit bus. The address is valid at the start of **T1** and remains so until the end of **T4**.

6.1.2 MemD0–31

The non-multiplexed data bus is 32 bits wide. The data bus is high impedance except when the transputer is writing data. If only one byte is being written, the unused 24 bits of the bus are high impedance at that time.

If the data setup time for read or write is too short it can be extended by inserting wait states at the end of **T2**.

6.1.3 notMemCE

The active low signal **notMemCE** is used to enable external memory on both read and write cycles.

		T801-20		T801-25			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Note**
TPCHEL	notMemCE falling from ProcClockOut rising	10	14	8	12	ns	
TPCLEH	ProcClockOut falling to notMemCE rising	10	14	8	12	ns	

Table 6.2 **notMemCE** to **ProcClockOut** skew

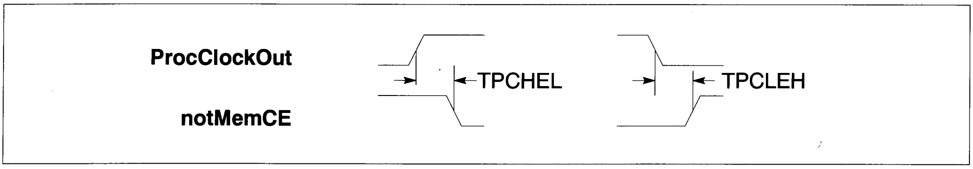


Figure 6.1 IMS T801 skew of **notMemCE** to **ProcClockOut**

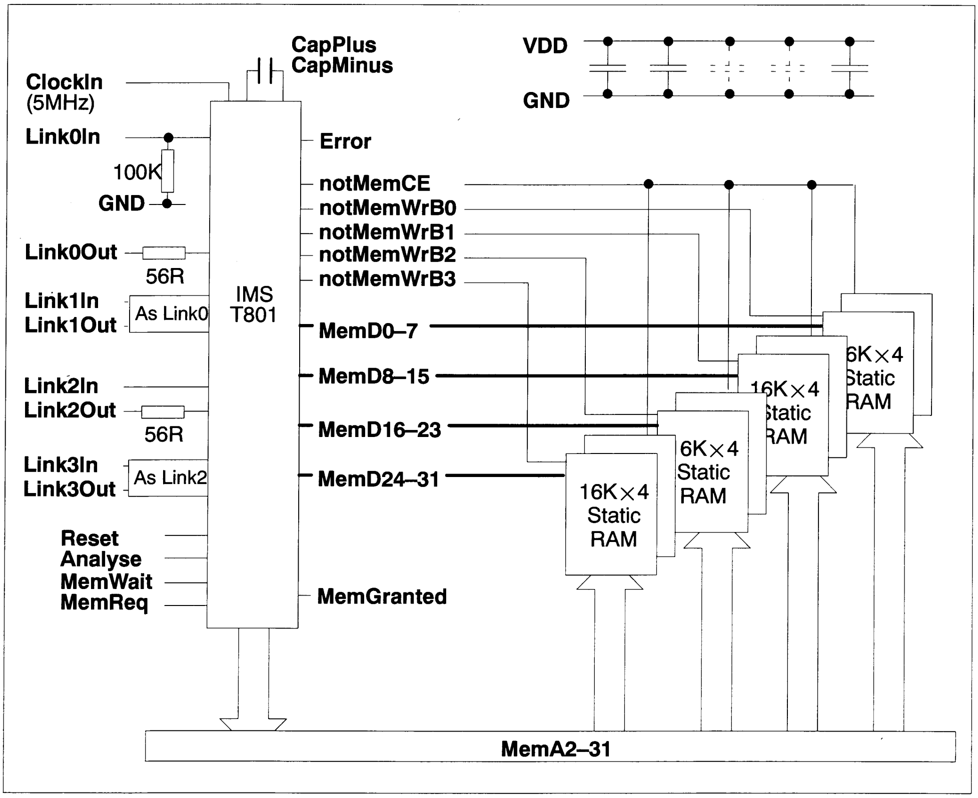


Figure 6.2 IMS T801 static RAM application

6.1.4 notMemWrB0–3

Four write enables **notMemWrB0-3** are provided, one to write each byte of a word. When writing a word, the four appropriate write enables are asserted; when writing a byte only the appropriate write enable is asserted.

6.1.5 MemWait

Wait states can be selected by taking **MemWait** high. Externally generated wait states of one complete processor cycle can be added to extend the duration of **T2** indefinitely.

6.1.6 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high.

MemGranted follows the timing of the bus being tristated and can be used to signal to the device requesting the DMA that it has control of the bus. Note that **MemGranted** changes on the falling edge of **ProcClockOut** and can therefore be sampled to establish control of the bus on the rising edge of **ProcClockOut**.

6.1.7 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn**. Its period is equal to one internal microcode cycle time, and can be derived from the formula

TPCLPCL = TDCLDCL / PLLx

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (Ordering section).

Edges of the various external memory strobes are synchronised by, but do not all coincide with, rising or falling edges of **ProcClockOut**.

Symbol	Parameter	T801-20		T801-25		Units	Notes
		Min	Max	Min	Max		
TPCLPCL	**ProcClockOut** period	49	51	39	41	ns	2
TPCHPCL	**ProcClockOut** pulse width high	22.5	27.5	17.5	22.5	ns	2
TPCLPCH	**ProcClockOut** pulse width low	**a**		**a**		ns	2, 3, 4
TPCstab	**ProcClockOut** stability		4		4	%	1, 2

Notes

1 Stability is the variation of cycle periods between two consecutive cycles, measured at corresponding points on the cycles.

2 This parameter is sampled and not 100% tested.

3 **a** is TPCLPCL – TPCHPCL.

4 This is a nominal value.

Table 6.3 **ProcClockOut**

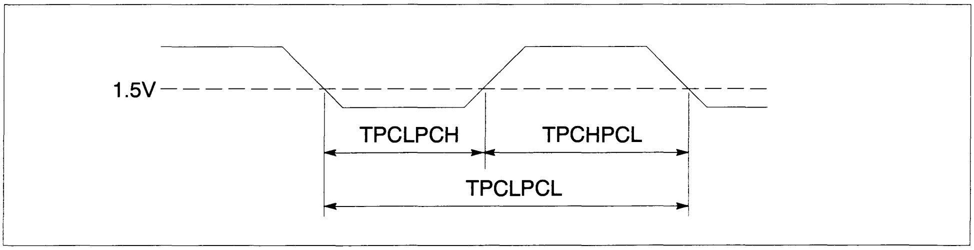


Figure 6.3 IMS T801 **ProcClockOut** timing

6.2 Read cycle

Read cycle data may be set up on the bus at any time after the start of **T1**, but must be valid when the IMS T801 reads it during **T4**. Data can be removed any time after the rising edge of **notMemCE**, but must be off the bus no later than the middle of **T1**, which allows for bus turn-around time before the data lines are driven at the start of **T2** in a processor write cycle.

Byte addressing is carried out internally by the IMS T801 for read cycles.

Symbol	Parameter	T801-20		T801-25		Units	Notes
		Min	Max	Min	Max		
TAVEL	Address valid before chip enable low	10		8		ns	1
TELEH	Chip enable low	72	78	58	64	ns	1
TEHEL	Delay before chip enable re-assertion	20		16		ns	1,2
TEHAX	Address hold after chip enable high	10		8		ns	1
TELDrV	Data valid from chip enable low	0	47	0	40	ns	
TAVDrV	Data valid from address valid	0	57	0	48	ns	
TDrVEH	Data setup before chip enable high	25		18		ns	
TEHDrZ	Data hold after chip enable high	0	20	0	16	ns	
TWEHEL	Write enable setup before chip enable low	20		16		ns	3
TPCHEL	ProcClockOut high to chip enable low	10		8		ns	1

Notes

1 This parameter is common to read and write cycles and to byte-wide memory accesses.

2 These values assume back-to-back external memory accesses.

3 Timing is for all four write enables **notMemWrB0-3**.

Table 6.4 Read cycle

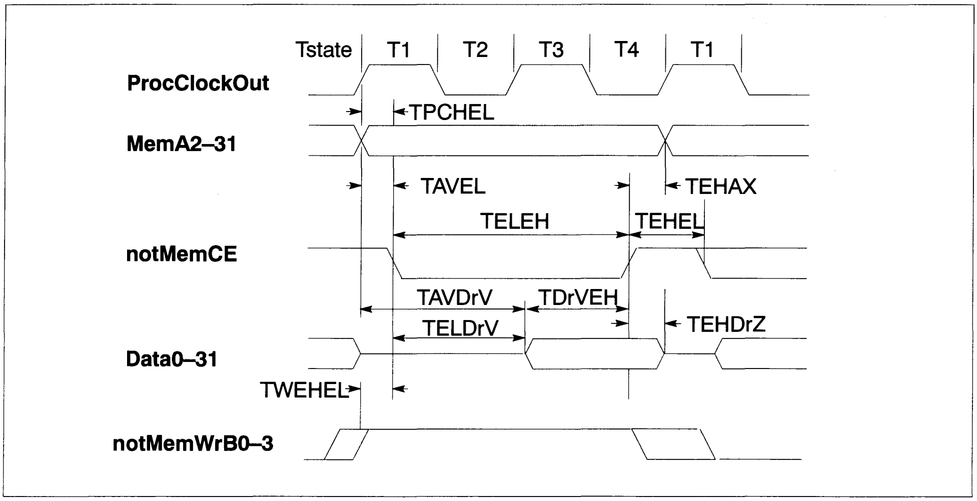


Figure 6.4 IMS T801 external read cycle

6.3 Write cycle

For write cycles the relevant bytes in memory are addressed by the write enables **notMemWrB0-3**. If a particular byte is not to be written, then the corresponding data outputs are tristated. **notMemWrB0** addresses the least significant byte.

The write enables are gated with the chip enable signal **notMemCE**, allowing them to be used without **notMemCE** for simple designs.

Data may be strobed into memory using **notMemWrB0-3** without the use of **notMemCE**, as the write enables go high between consecutive external memory write cycles.

Write data is placed on the data bus at the start of **T2** and removed at the end of **T4**. The write cycle is completed with **notMemCE** going high.

		T801-20		T801-25			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Unit**	**Note**
TDwVEH	Data setup before chip enable high	50		40		ns	1
TEHDwZ	Data hold after write	10	15	8	12	ns	1
TDwZEL	Write data invalid to next chip enable	10		8		ns	1
TWELEL	Write enable setup to chip enable low	–3	0	–2	0	ns	1,2
TEHWEH	Write enable hold after chip enable high	0	3	0	2	ns	1,2

Notes

1 Timing is for all four write enables **notMemWrB0-3**.

Table 6.5 Write cycle

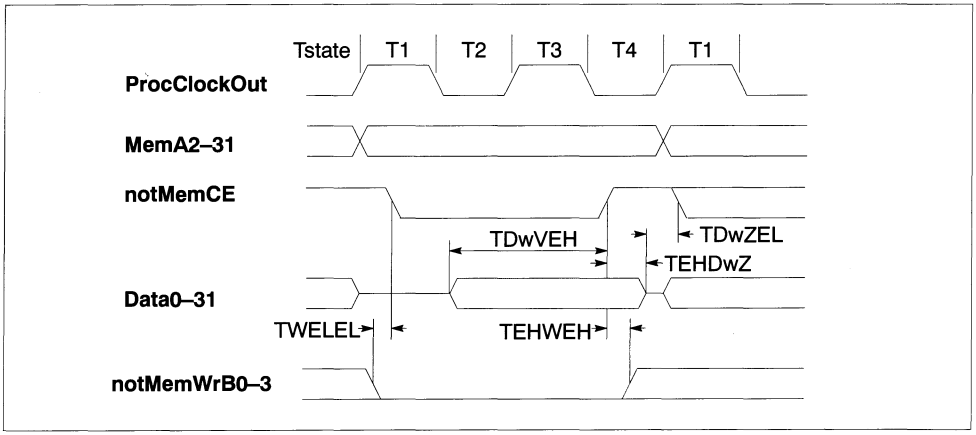


Figure 6.5 IMS T801 external write cycle

6.4 Wait

Taking **MemWait** high with the timing shown in the diagram will extend the duration of **T2** by one processor cycle **TPCLPCL**. One wait state comprises the pair **W1** and **W2**. **MemWait** is sampled during **T2**, and should not change state in this region. If **MemWait** is still high when sampled in **W2** then another wait period will be inserted. This can continue indefinitely. Internal memory access is unaffected by the number of wait states selected.

The wait state generator can be a simple digital delay line, synchronised to **notMemCE**. The **Single Wait State Generator** circuit in figure 6.7 can be extended to provide two or more wait states, as shown in figure 6.8.

Symbol	Parameter	T801–20		T801–25		Units	Notes
		Min	Max	Min	Max		
TPCHWtH	**MemWait** asserted after **ProcClockOut** high		25		20	ns	
TAVWtH	**MemWait** asserted after Address valid		25		20	ns	
TPCHWtL	**MemWait** low after **ProcClockOut** high	35		28			

Table 6.6 Memory wait

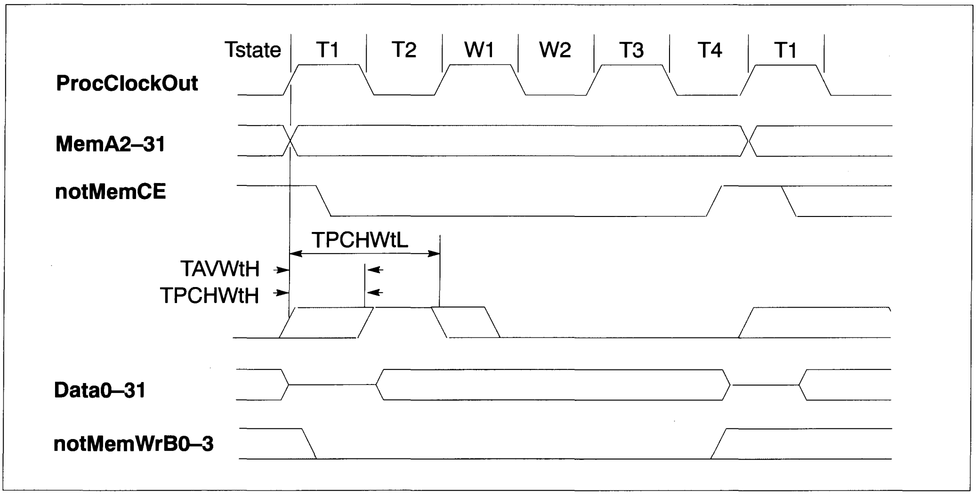


Figure 6.6 IMS T801 memory wait timing

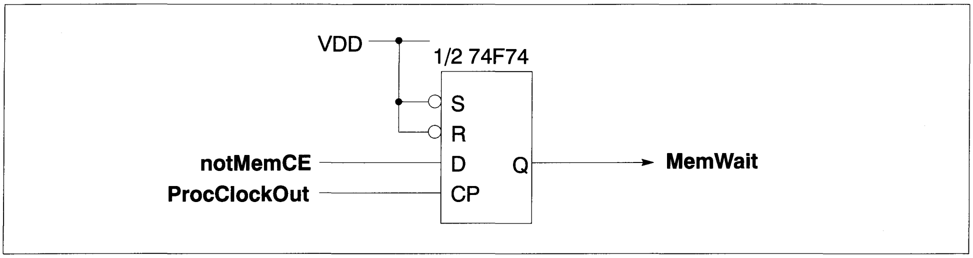


Figure 6.7 Single wait state generator

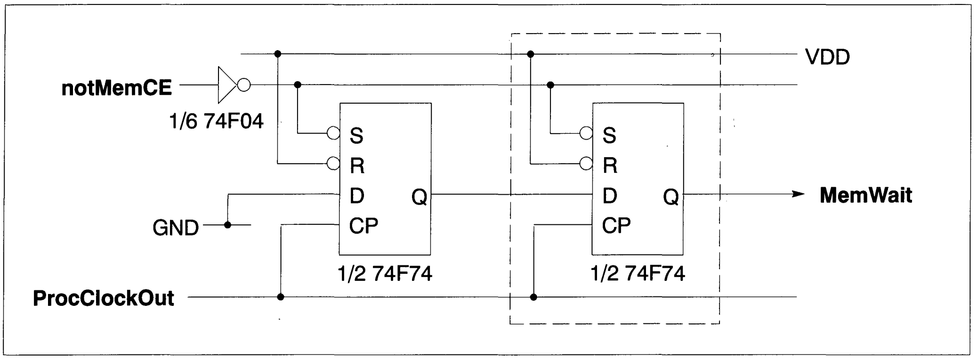


Figure 6.8 Extendable wait state generator

6.5 Direct memory access

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high. **MemReq** is sampled during **T1** of the processor cycle and the DMA device will then have control of the bus at the beginning of the next processor cycle, (after one **ProcClockOut** for internal accesses and two **ProcClockOut** cycles for external memory accesses, without wait states). When the processor transfers control of the bus the signals **MemA2-31**, **notMemWrB0-3** and **notMemCE** are tristated and **MemGranted** is asserted high. **MemGranted** follows the timing of the bus being tristated and can be used to signal to the device requesting the DMA that it has control of the bus. Note that **MemGranted** changes on the falling edge of **ProcClockOut** and can therefore be sampled to establish control of the bus on the rising edge of **ProcClockOut**. During the DMA cycles, **MemReq** is sampled during each high phase of **ProcClockOut** and after it is taken low, control of the bus will be returned to the processor within two **ProcClockOut** cycles.

The processor is still able to access its internal memory while the DMA transfer proceeds, however when an external memory request is made the processor is forced to wait until the end of the DMA request. The DMA device has no access to the transputer's internal memory.

While control of the bus is being transferred from the processor to the DMA device, an extra clock phase, (one quarter of a **ProcClockOut** cycle) is allowed before the DMA transfer begins to ensure that the **notMemCE** and **notMemWrB0-3** signals have been driven high before being tristated. This normally removes the requirement for external pull-up resistors.

DMA allows a bootstrap program to be loaded into external memory for execution after reset. If **MemReq** is asserted high during reset, **MemGranted** will be asserted high allowing access to the external memory before the bootstrap sequence begins. **MemReq** must be asserted for at least one period of **TDCLDCL** of **ClockIn** before Reset is asserted. The DMA control circuitry should ensure that correct operation will result if **Reset** should interrupt a normal DMA cycle.

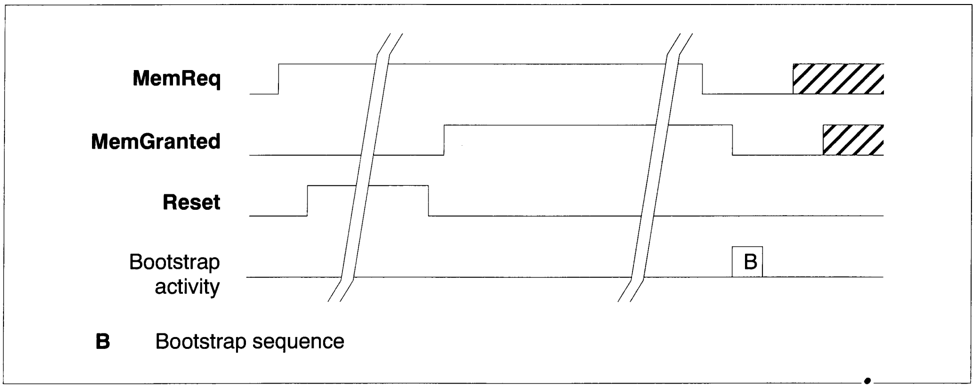


Figure 6.9 IMS T801 DMA sequence at reset

Symbol	Parameter	T801-20		T801-25		Units	Notes
		Min	Max	Min	Max		
TMRHMGH	**MemReq** response time	85	**a**	70	**a**	ns	1
TMRLMGL	**MemReq** end response time	90	100	75	80	ns	
TAZMGH	Address bus tristate before **MemGranted**	0		0		ns	
TDZMGH	Data bus tristate before **MemGranted**	0		0		ns	

Notes

1 Maximum response time **a** depends on whether an external memory cycle is in progress. Maximum time is (2 processor cycles) + (number of wait state cycles) for word access.

Table 6.7 Memory request

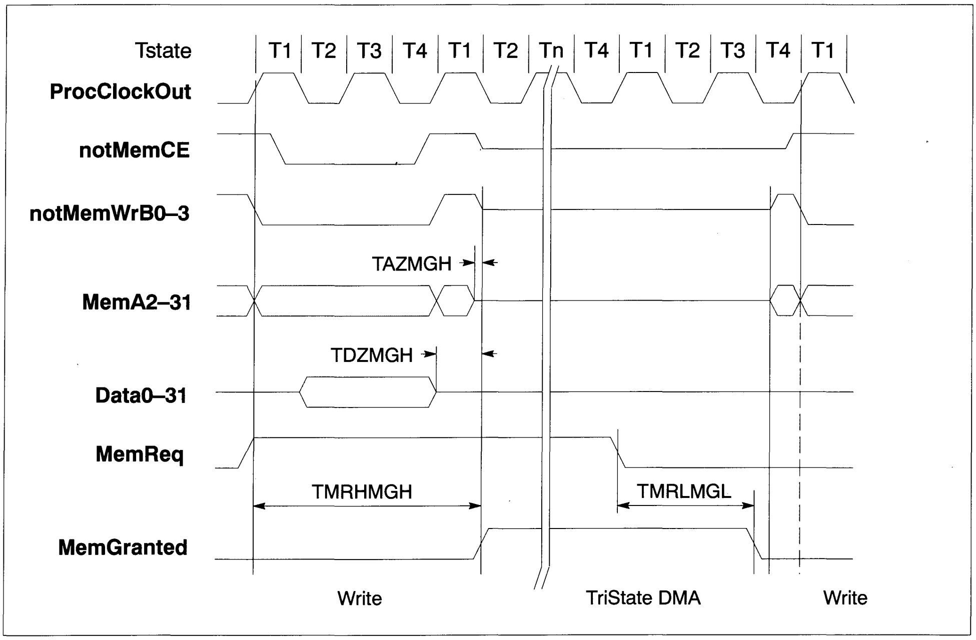


Figure 6.10 Memory request timing

7 Events

EventReq and **EventAck** provide an asynchronous handshake interface between an external event and an internal process. When an external event takes **EventReq** high the external event channel (additional to the external link channels) is made ready to communicate with a process. When both the event channel and the process are ready the processor takes **EventAck** high and the process, if waiting, is scheduled. **EventAck** is removed after **EventReq** goes low.

EventWaiting is asserted high by the transputer when a process executes an input on the event channel; typically with the occam `EVENT ? ANY` instruction. It remains high whilst the transputer is waiting for or servicing **EventReq** and is returned low when **EventAck** goes high. The **EventWaiting** pin changes near the falling edge of **ProcClockOut** and can therefore be sampled by the rising edge of **ProcClockOut**.

The **EventWaiting** pin can only be asserted by executing an *in* instruction on the event channel. The **EventWaiting** pin is not asserted high when an enable channel (*enbc*) instruction is executed on the Event channel (during an ALT construct in occam, for example). The **EventWaiting** pin can be asserted by executing the occam input on the event channel (such as `Event ? ANY`), provided that this does not occur as a guard in an alternative process. The **EventWaiting** pin can not be used to signify that an alternative process (ALT) is waiting on an input from the event channel.

EventWaiting allows a process to control external logic; for example, to clock a number of inputs into a memory mapped data latch so that the event request type can be determined.

Only one process may use the event channel at any given time. If no process requires an event to occur **EventAck** will never be taken high. Although **EventReq** triggers the channel on a transition from low to high, it must not be removed before **EventAck** is high. **EventReq** should be low during **Reset**; if not it will be ignored until it has gone low and returned high. **EventAck** is taken low when **Reset** occurs.

If the process is a high priority one and no other high priority process is running, the latency is as described on page 32. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

Cycle 1	Sample **EventReq** at pad on the rising edge of **ProcClockOut** and synchronise.
Cycle 2	Edge detect the synchronised **EventReq** and form the interrupt request.
Cycle 3	Sample interrupt vector for microcode ROM in the CPU.
Cycle 4	Execute the interrupt routine for Event rather than the next instruction.

Symbol	Parameter	T801-20		T801-25		Units	Notes
		Min	Max	Min	Max		
TVHKH	Event request response	0		0		ns	1
TKHVL	Event request hold	0		0		ns	1
TVLKL	Delay before removal of event acknowledge	0	158	0	128	ns	
TKLVH	Delay before re-assertion of event request	0		0		ns	1
TKHEWL	Event acknowledge to end of event waiting	0		0		ns	1

Notes

1 Guaranteed, but not tested.

Table 7.1 Event

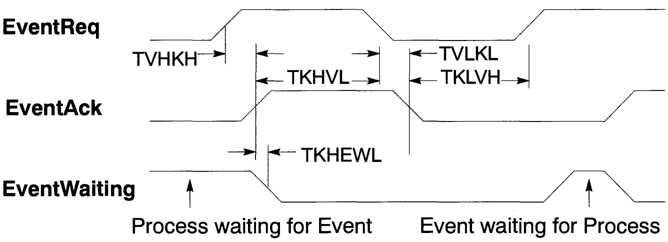


Figure 7.1 IMS T801 event timing

8 Links

Four identical INMOS bi-directional serial links provide synchronized communication between processors and with the outside world. Each link comprises an input channel and output channel. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

The IMS T801 links allow an acknowledge packet to be sent before the data packet has been fully received. This overlapped acknowledge technique is fully compatible with all other INMOS transputer links.

The IMS T801 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 20 Mbits/sec for IMS T801-20 and IMS T801-25. Links are not synchronised with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 Ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

Link speeds can be set by **LinkSpeed**. **LinkSpeed** allows Links 0, 1, 2 or 3 to be set to 10 or 20 Mbits/sec. Table 8.1 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

Link		Kbytes/sec	
Speed	Mbits/sec	Uni	Bi
0	10	910	1250
1	20	1740	2350

Table 8.1 Speed Settings for Transputer Links

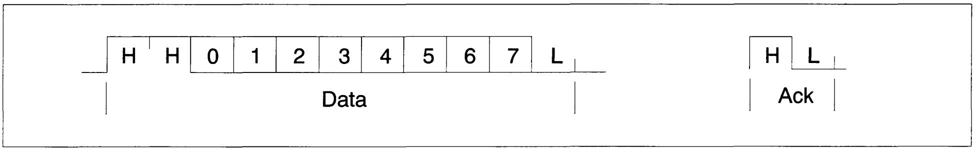


Figure 8.1 IMS T801 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	1
TJDf	**LinkIn** fall time				20	ns	1
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100W transmission line			56		ohms	

Notes

1 Guaranteed, but not tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 8.2 Link

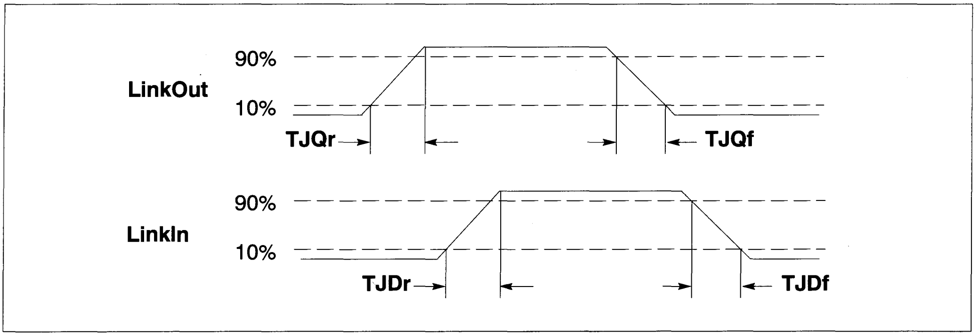


Figure 8.2 IMS T801 link timing

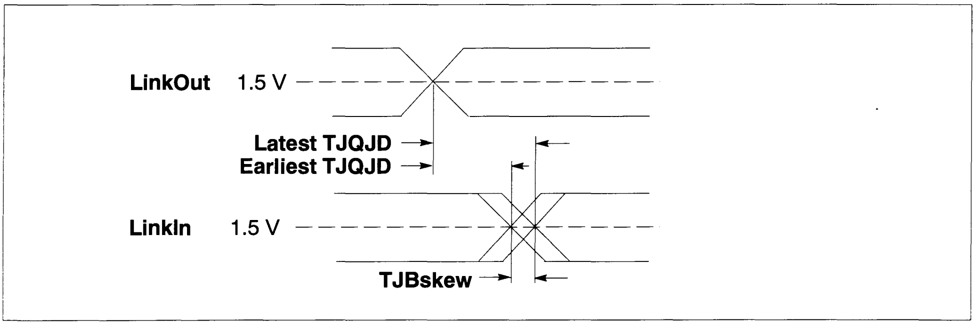


Figure 8.3 IMS T801 buffered link timing

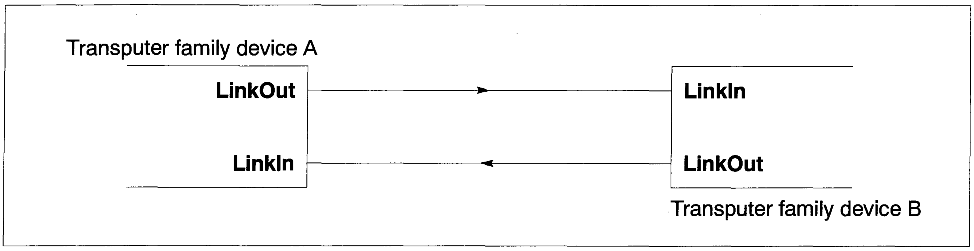


Figure 8.4 Links directly connected

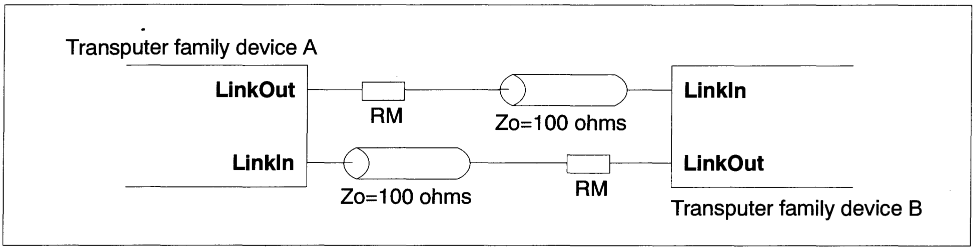


Figure 8.5 Links connected by transmission line

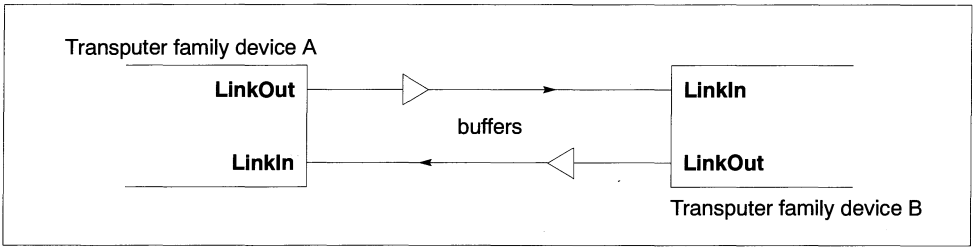


Figure 8.6 Links connected by buffers

9 Electrical specifications

9.1 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
II	Input current		±25	mA	4
tOSC	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 9.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1,2
CL	Load capacitance on any pin		60	pF	3
TA	Operating temperature range	0	70	°C	4

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Excluding **LinkOut** load capacitance.

4 Air flow rate 400 linear ft/min transverse air flow.

Table 9.2 Operating conditions

SYMBOL	PARAMETER		MIN	MAX	UNITS	NOTES
VIH	High level input voltage		2.0	VDD+0.5	V	1, 2
VIL	Low level input voltage		–0.5	0.8	V	1, 2
II	Input current	@ GND<VI<VDD		±10	µA	1, 2
VOH	Output high voltage	@ IOH=2mA	VDD–1		V	1, 2
VOL	Output low voltage	@ IOL=4mA		0.4	V	1, 2
IOZ	Tristate output current	@ GND<V0<VDD		±10	µA	1, 2
PD	Power dissipation			1.2	W	2, 3
CIN	Input capacitance	@ f=1MHz		7	pF	4
COZ	Output capacitance	@ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

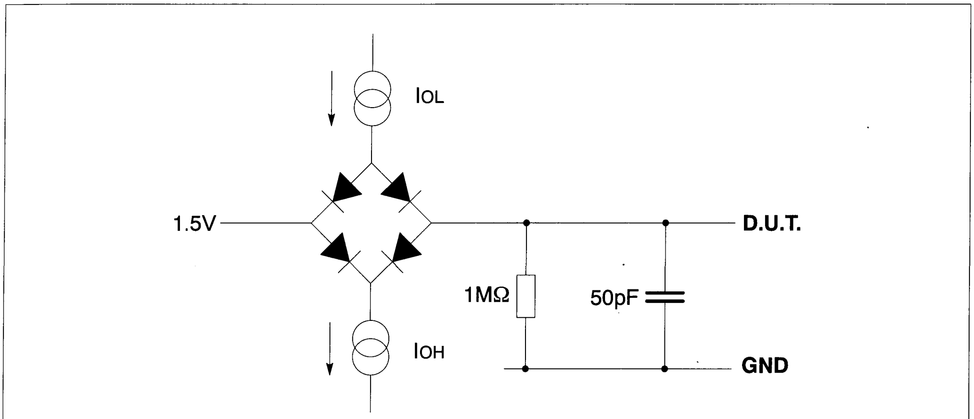
2 Parameters for IMS T801-S measured at 4.75V<**VDD**<5.25V and 0^{o}C<**TA**<70^{o}C.
Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution.
Power dissipation for processor operating at 20 MHz.

4 This parameter is sampled and not 100% tested.

Table 9.3 DC characteristics

9.2 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 9.1 Load circuit for AC measurements

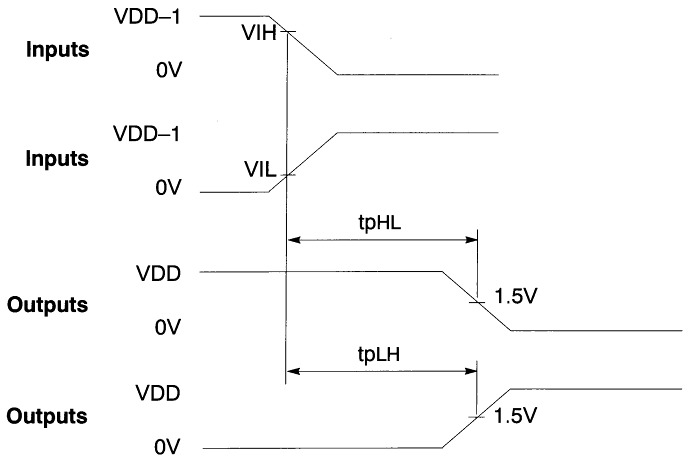


Figure 9.2 AC measurements timing waveforms

9.3 AC timing characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTE
TDr	Input rising edges	2	20	ns	1,2,3
TDf	Input falling edges	2	20	ns	1,2,3
TQr	Output rising edges		25	ns	1,3
TQf	Output falling edges		15	ns	1,3

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 Guaranteed, but not tested.

Table 9.4 Input and output edges

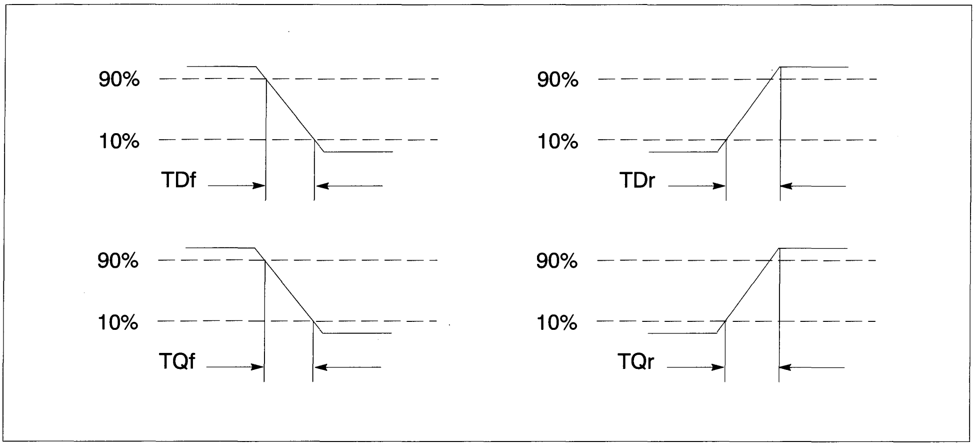


Figure 9.3 IMS T801 input and output edge timing

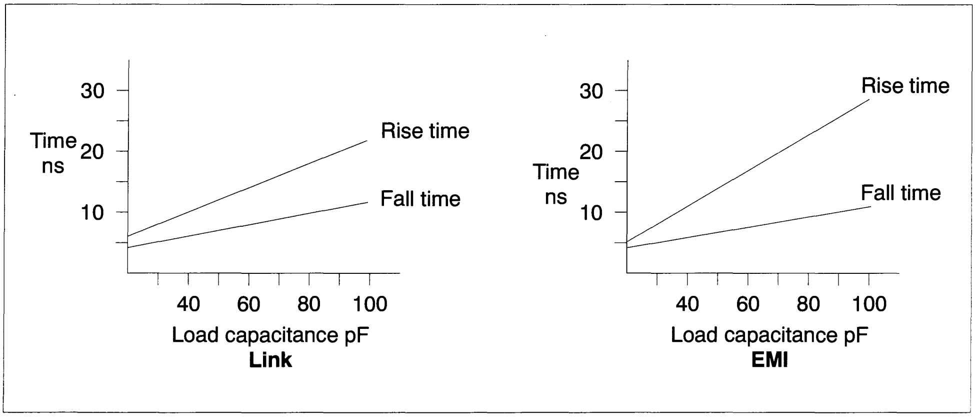


Figure 9.4 Typical rise/fall times

9.4 Power rating

Internal power dissipation P_{INT} of transputer and peripheral chips depends on **VDD**, as shown in figure 9.5. P_{INT} is substantially independent of temperature.

Total power dissipation P_D of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + q\, J_A * P_D$$

where T_A is the external ambient temperature in °C and θJ_A is the junction-to-ambient thermal resistance in °C/W. θJ_A for each package is given in Appendix A, Packaging Specifications.

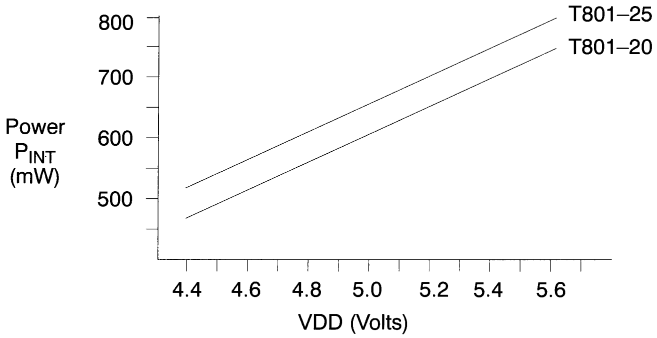


Figure 9.5 IMS T801 internal power dissipation vs VDD

10 Package pinouts

10.1 100 pin grid array package

Index	1	2	3	4	5	6	7	8	9	10
A	A21	A23	A25	A26	A30	A31	D2	D5	D6	D13
B	A5	A9	A11	A24	A29	GND	D3	D7	VDD	D14
C	A4	A6	A8	A22	A10	D0	D4	D9	D12	D16
D	GND	A2	A3	A7	A27	D1	D8	D10	D15	D17
E	A17	A19	A18	A20	A28	D11	D18	D19	D20	D21
F	A16	A15	A14	A13	Reset	Error Out	D24	GND	D23	D22
G	A12	not Mem WrB2	not Mem WrB0	GND	Link In1	Link Speed	D31	D27	D26	D25
H	not Mem WrB3	Mem Wait	Mem Req	Link Out3	Link In0	Proc Clock Out	GND	Proc Speed Select1	D30	D28
J	not Mem WrB1	Mem Granted	Event Req	Link In2	Link Out1	Event Waiting	ClockIn	Boot From ROM	Analyse	D29
K	not Mem CE	Event Ack	Link In3	Link Out2	Link Out0	VDD	Cap Plus	Cap Minus	Proc Speed Select0	Proc Speed Select2

Figure 10.1 IMS T801 100 pin grid array package pinout – top view

11 Ordering

This section indicates the designation of speed and package selections for the various devices. Speed of **ClockIn** is 5 MHz for all parts. Transputer processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in the external memory section.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	Processor clock speed	Processor cycle time	PLLx	Package
IMS T801-G20S	20.0	50	4.0	100 pin ceramic pin grid array
IMS T801-G25S	25.0	40	5.0	100 pin ceramic pin grid array

Table 11.1 IMS T801 ordering details

IMS T426 transputer

inmos®

Preliminary Data

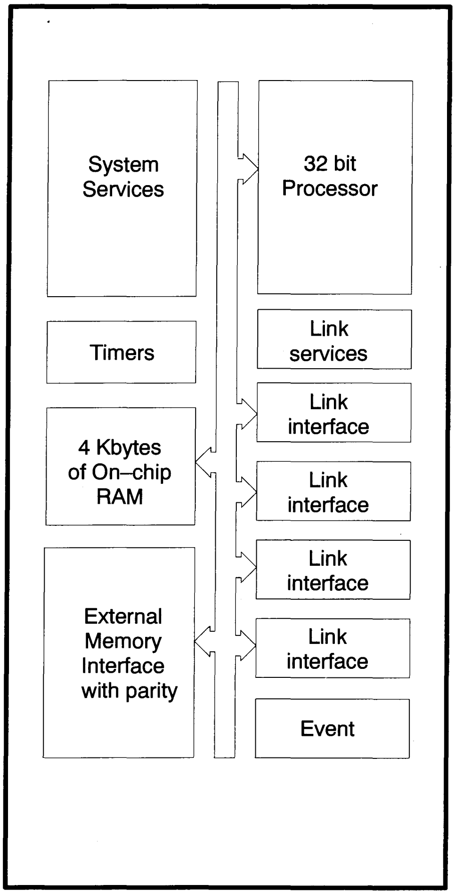
The information in this datasheet is subject to change

FEATURES

32 bit architecture
40 ns internal cycle time
25 MIPS peak instruction rate
Debugging support
4 Kbytes on–chip static RAM
100 Mbytes/sec sustained data rate to internal memory
4 Gbytes directly addressable external memory
33 Mbytes/sec sustained data rate to external memory with parity
750 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
High performance graphics support with block move instructions
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ± 5% power supply
Packaging 100 pin CQFP

APPLICATIONS

High speed multi processor systems
High performance graphics processing
Supercomputers
Workstations and workstation clusters
Digital signal processing
Accelerator processors
Distributed databases
System simulation
Telecommunications
Robotics
Fault tolerant systems
Image processing
Pattern recognition
Artificial intelligence
Disk systems



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992
42 1509 01

1 Introduction

The IMS T426 transputer is a 32 bit CMOS microcomputer with graphics support. It has 4 Kbytes on-chip RAM for high speed processing, a configurable memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages and provides direct support for the occam model of concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond.

For convenience of description, the IMS T426 operation is split into the basic blocks shown in figure 1.1.

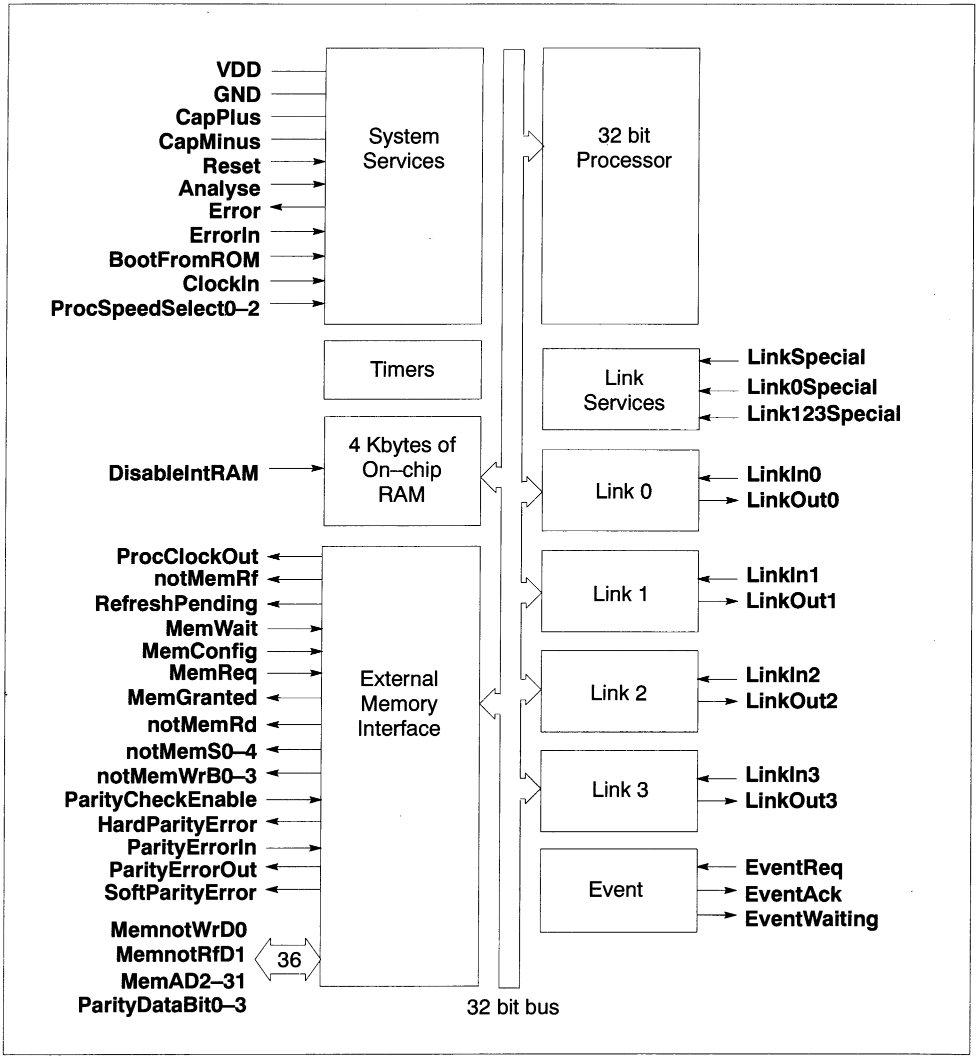


Figure 1.1 IMS T426 block diagram

The processor speed of a device can be pin-selected in stages from 20 MHz up to the maximum allowed for the part. A device running at 25 MHz achieves an instruction throughput of 25 MIPS peak and 12.5 MIPS sustained.

High performance graphics support is provided by microcoded block move instructions which operate at the speed of memory. The two-dimensional block move instructions provide for contiguous block moves as well as block copying of either non-zero bytes of data only or zero bytes only. Block move instructions can be used to provide graphics operations such as text manipulation, windowing, panning, scrolling and screen updating.

Cyclic redundancy checking (CRC) instructions are available for use on arbitrary length serial data streams, to provide error detection where data integrity is critical. Another feature of the IMS T426, useful for pattern recognition, is the facility to count bits set in a word.

The IMS T426 can directly access a linear address space of 4 Gbytes. The memory interface uses multiplexed data and address lines and provides a data rate of up to 4 bytes every 120 nanoseconds (33 Mbytes/sec) for a 25 MHz device. In addition to the 32 bit multiplexed data and address bus there are four I/O parity data lines, providing a data check feature for each byte. A configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems.

System Services include processor reset and bootstrap control, together with facilities for error analysis. Error signals may be daisy-chained in multi-transputer systems.

The standard INMOS communication links allow networks of transputer family products to be constructed by direct point to point connections with no external logic. The IMS T426 links support the standard operating speed of 10 Mbits/sec, but also operate at 5 or 20 Mbits/sec. Each link can transfer data bi-directionally at up to 2.35 Mbytes/sec.

The transputer is designed to implement the occam language, detailed in the *occam Reference Manual*, but also efficiently supports other languages such as C, Pascal and Fortran. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*. The instruction set of the IMS T426 is a superset of the IMS T800, excluding the FPU instructions. Additional instructions are listed in Chapter 4.

This datasheet supplies hardware implementation and characterization details for the IMS T426. It is intended to be read in conjunction with the Transputer Architecture chapter, which details the architecture of the transputer and gives an overview of occam.

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high.
Pinout details are given on page 216.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
ProcSpeedSelect0–2	in	Processor speed selectors
Reset	in	System reset
Error	out	Error indicator
ErrorIn	in	Error daisychain input
Analyse	in	Error analysis
BootFromROM	in	Boot from external ROM or from link
DisableIntRAM	in	Disable internal RAM

Table 2.1 IMS T426 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemnotWrD0	in/out	Multiplexed data bit 0 and write cycle warning
MemnotRfD1	in/out	Multiplexed data bit 1 and refresh warning
MemAD2–31	in/out	Multiplexed data and address bus
ParityDataBit0–3	in/out	Parity state on data bus
ParityCheckEnable	in	Enable parity checking
HardParityError	out	Parity error indicator after a retry cycle fails
SoftParityError	out	Parity error indicator after a successful retry cycle
ParityErrorIn	in	Parity error daisychain input
ParityErrorOut	out	Parity error daisychain output
notMemS0–4	out	Five general purpose strobes
notMemWrB0–3	out	Four byte–addressing write strobes
MemWait	in	Memory cycle extender
MemConfig	in	Memory configuration data input
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted
notMemRd	out	Read strobe
notMemRf	out	Dynamic memory refresh indicator
RefreshPending	out	Dynamic memory refresh cycle is pending

Table 2.2 IMS T426 programmable memory interface

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge
EventWaiting	out	Event input requested by software

Table 2.3 IMS T426 event

Pin	In/Out	Function
LinkIn0–3	in	Four serial data input channels
LinkOut0–3	out	Four serial data output channels
LinkSpecial	in	Select non–standard speed as 5 or 20 Mbits/sec
Link0Special	in	Select special speed for Link 0
Link123Special	in	Select special speed for Links 1, 2, 3

Table 2.4 IMS T426 link

3 System services

System services include all the necessary logic to initialize and sustain operation of the device. They also include error handling and analysis facilities.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimize inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1 μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 10 MHz. If a polarized capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

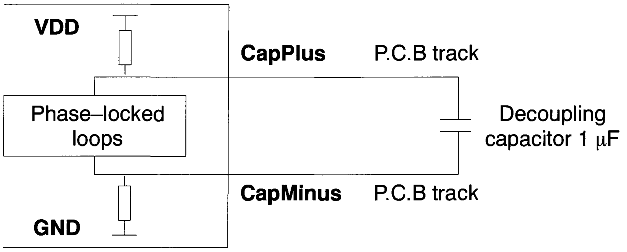


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	T426-20, -25			Units	Notes
		Min	Nom	Max		
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1, 2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1, 3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1, 4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 8.3).

Table 3.1 Input clock

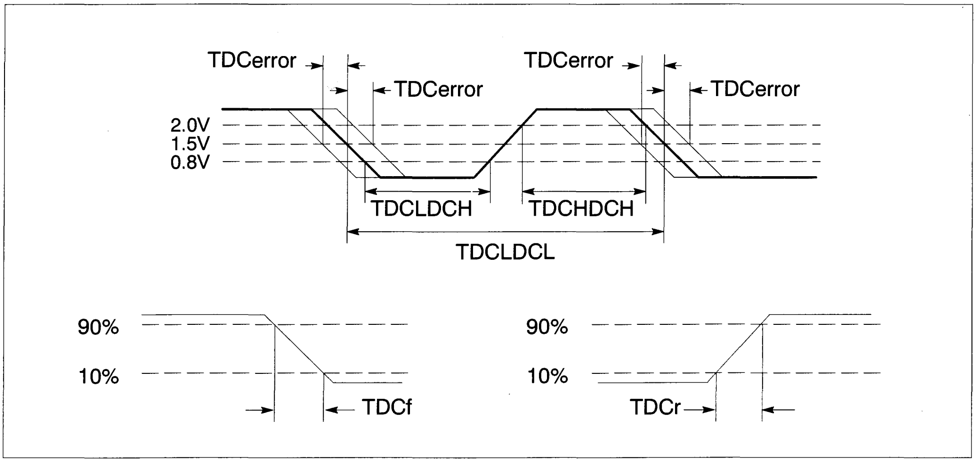


Figure 3.2 **ClockIn** timing

3.4 ProcSpeedSelect0–2

Processor speed of the IMS T426 is variable in discrete steps. The desired speed can be selected, up to the maximum rated for a particular component, by the three speed select lines **ProcSpeedSelect0-2**. The pins are tied high or low, according to table 3.2, for the various speeds.

Only six of the possible speed select combinations are currently used; the other two are not valid speed selectors. The frequency of **ClockIn** for the speeds given in the table is 5 MHz.

ProcSpeed-Select2	ProcSpeed-Select1	ProcSpeed-Select0	Processor Clock Speed MHz	Processor Cycle Time ns	Notes
0	0	0	20.0	50.0	
0	0	1	22.5	44.4	Not supported
0	1	0	25.0	40.0	
0	1	1	30.0	33.3	Not supported
1	0	0	35.0	28.6	Not supported
1	0	1			Invalid
1	1	0	17.5	57.1	Not supported
1	1	1			Invalid

Table 3.2 Processor speed selection

3.5 Bootstrap

The transputer can be bootstrapped either from a link or from external ROM. To facilitate debugging, **BootFromROM** may be dynamically changed but must obey the specified timing restrictions. It is sampled once only by the transputer, before the first instruction is executed after **Reset** is taken low.

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFFFFFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the **W** register points to **MemStart** (page 174).

If **BootFromROM** is connected low (e.g. to **GND**) the transputer will wait for the first bootstrap message to arrive on any one of its links. The transputer is ready to receive the first byte on a link within two processor cycles **TPCLPCL** after **Reset** goes low.

If the first byte received (the control byte) is greater than 1 it is taken as the quantity of bytes to be input. The following bytes, to that quantity, are then placed in internal memory starting at location **MemStart**. Following reception of the last byte the transputer will start executing code at **MemStart** as a low priority process. **BootFromROM** may be taken high after reception of the last byte, if required. The memory space immediately above the loaded code is used as work space. A byte arriving on other links after the control byte has been received and on the bootstrapping link after the last bootstrap byte, will be retained and no acknowledge will be sent until a process inputs from them.

3.6 Peek and poke

Any location in internal or external memory can be interrogated and altered when the transputer is waiting for a bootstrap from link. If the control byte is 0 then eight more bytes are expected on the same link. The first four byte word is taken as an internal or external memory address at which to *poke* (write) the second four byte word. If the control byte is 1 the next four bytes are used as the address from which to *peek* (read) a word of data; the word is sent down the output channel of the same link.

Following such a *peek* or *poke*, the transputer returns to its previously held state. Any number of accesses may be made in this way until the control byte is greater than 1, when the transputer will commence reading its bootstrap program. Any link can be used, but addresses and data must be transmitted via the same link as the control byte.

3.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initializes the transputer, triggers the memory configuration sequence and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) is ignored whilst **Reset** is high but can occur before bootstrap (page 195).

After the end of **Reset** there will be a delay of 144 periods of **ClockIn** (figure 3.3). Following this, the **MemnotWrD0**, **MemnotRfD1** and **MemAD2-31** pins will be scanned to check for the existence of a pre-programmed memory interface configuration (page 198). This lasts for a further 144 periods of **ClockIn**. Regardless of whether a configuration was found, 36 configuration read cycles will then be performed on external memory using the default memory configuration (page 201), in an attempt to access the external configuration ROM. A delay will then occur, its period depending on the actual configuration. Finally eight complete and consecutive refresh cycles will initialize any dynamic RAM, using the new memory configuration. If the memory configuration does not enable refresh of dynamic RAM the refresh cycles will be replaced by an equivalent delay with no external memory activity.

If **BootFromROM** is high bootstrapping will then take place immediately, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

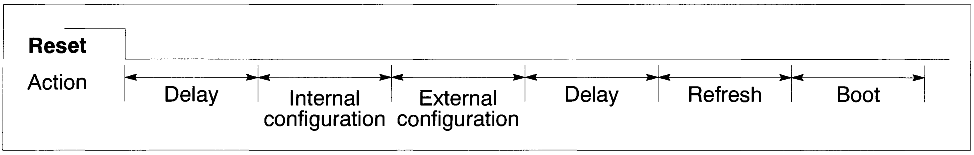


Figure 3.3 IMS T426 post–reset sequence

3.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 48). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags **Error**, **HaltOnError** and **EnableJ0Break** are normally cleared at reset on the IMS T426; however, if **Analyse** is asserted the flags are not altered. Memory refresh continues.

Input links will continue with outstanding transfers. Output links will not make another access to memory for data but will transmit only those bytes already in the link buffer. Providing there is no delay in link acknowledgement, the links should be inactive within a few microseconds of the transputer halting.

Reset should not be asserted before the transputer has halted and link transfers have ceased. When **Reset** is taken low whilst **Analyse** is high, neither the memory configuration sequence nor the block of eight refresh cycles will occur; the previous memory configuration will be used for any external memory accesses. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 3.3.

I	**MemStart** if bootstrapping from a link, or the external memory bootstrap address if bootstrapping from ROM.
W	**MemStart** if bootstrapping from ROM, or the address of the first free word after the bootstrap program if bootstrapping from link.
A	The value of **I** when the processor halted.
B	The value of **W** when the processor halted, together with the priority of the process when the transputer was halted (i.e. the **W** descriptor).
C	The ID of the bootstrapping link if bootstrapping from link.

Table 3.3 Register values after **Analyse**

Symbol	Parameter	T426-20, -25			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

1. Full periods of **ClockIn TDCLDCL** required.
2. At power-on reset.
3. Must be stable until after end of bootstrap period. See Bootstrap section 3.5.

Table 3.4 **Reset** and **Analyse** timing

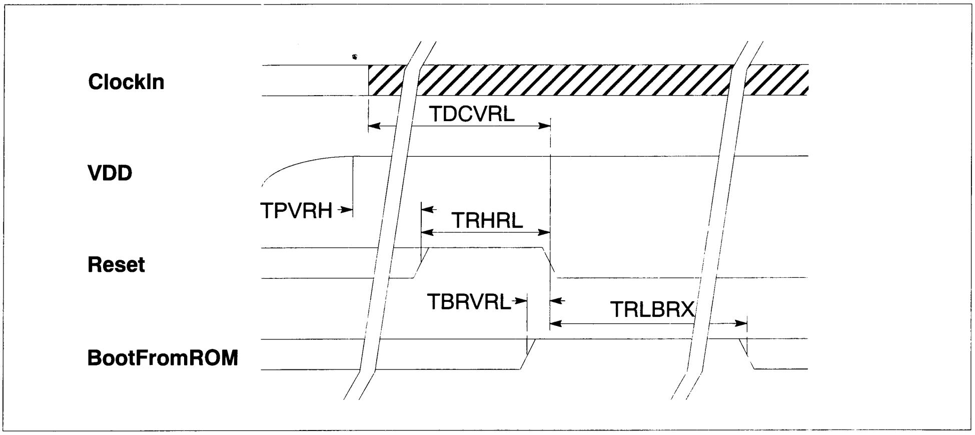


Figure 3.4 Transputer **Reset** timing with **Analyse** low

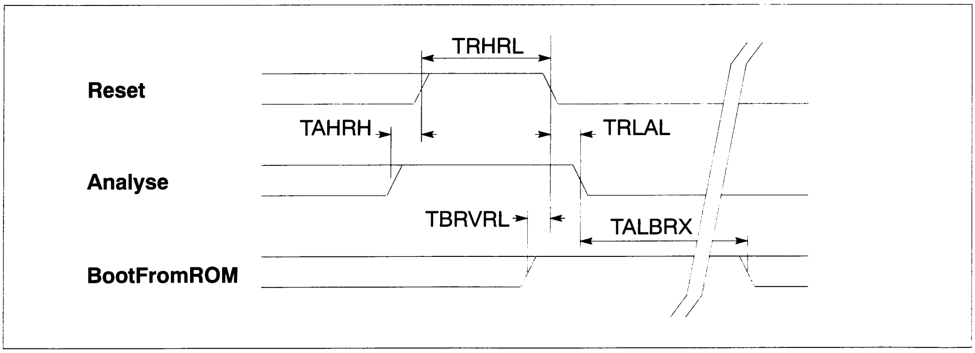


Figure 3.5 **Reset** and **Analyse** timing

3.9 Error, ErrorIn

The **Error** pin carries the OR'ed output of the internal **Error** flag and the **ErrorIn** input. If **Error** is high it indicates either that **ErrorIn** is high or that an error was detected in one of the processes. An internal error can be caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 48). Once set, the **Error** flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (section 3.8).

A process can be programmed to stop if the **Error** flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data. **ErrorIn** does not directly affect the status of a processor in any way.

By setting the **HaltOnError** flag the transputer itself can be programmed to halt if **Error** becomes set. If **Error** becomes set after **HaltOnError** has been set, all processes on that transputer will cease but will not necessarily cause other transputers in a network to halt. Setting **HaltOnError** after **Error** will not cause the transputer to halt; this allows the processor reset and analyse facilities to function with the flags in indeterminate states.

An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by 'daisy-chaining' the **ErrorIn** and **Error** pins of a number of processors and applying the final **Error** output signal to the **EventReq** pin of a suitably programmed master transputer (see figure 3.6). Since the process state is preserved when stopped by an error, the master transputer can then use the analyse function to debug the fault. When using such a circuit, note that the **Error** flag is in an indeterminate state on power up; the circuit and software should be designed with this in mind.

Error checks can be removed completely to optimize the performance of a proven program; any unexpected error then occurring will have an arbitrary undefined effect.

If a high priority process pre-empts a low priority one, status of the **Error** and **HaltOnError** flags is saved for the duration of the high priority process and restored at the conclusion of it. Status of both flags is transmitted to the high priority process. Either flag can be altered in the process without upsetting the error status of any complex operation being carried out by the pre-empted low priority process.

In the event of a transputer halting because of **HaltOnError**, the links will finish outstanding transfers before shutting down. If **Analyse** is asserted then all inputs continue but outputs will not make another access to memory for data. Memory refresh will continue to take place.

After halting due to the **Error** flag changing from 0 to 1 whilst **HaltOnError** is set, register **I** points two bytes past the instruction which set **Error**. After halting due to the **Analyse** pin being taken high, register **I** points one byte past the instruction being executed. In both cases **I** will be copied to register **A**.

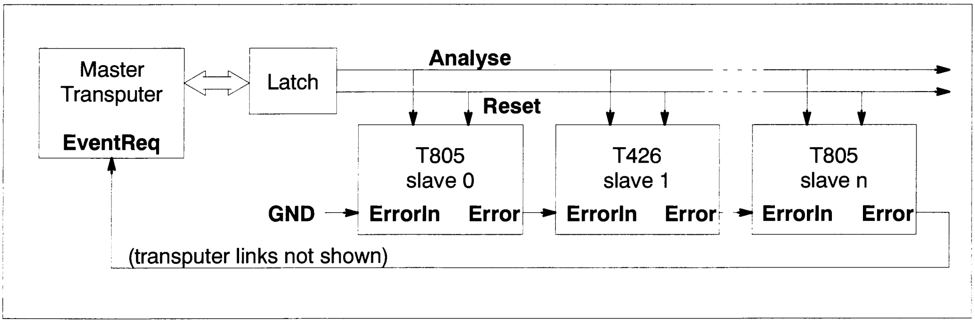


Figure 3.6 Error handling in a multi–transputer system

4 Memory

The IMS T426 has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (page 178). The transputer can also access 4 Gbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

IMS T426 memory is byte addressed, with words aligned on four-byte boundaries. The least significant byte of a word is the lowest addressed byte. The bits in a byte are numbered 0 to 7, with bit 0 the least significant. The bytes are numbered from 0, with byte 0 the least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

Internal memory starts at the most negative address #80000000 and extends to #80000FFF. User memory begins at #80000070; this location is given the name **MemStart**. An instruction *ldmemstartval* is provided to obtain the value of **MemStart**.

The context of a process in the transputer model involves a workspace descriptor (**WPtr**) and an instruction pointer (**IPtr**). **WPtr** is a word address pointer to a workspace in memory. **IPtr** points to the next instruction to be executed for the process which is the currently executing process. The context switch performed by the breakpoint instruction swaps the **WPtr** and **IPtr** of the currently executing process with the **WPtr** and **IPtr** held above **MemStart**. Two contexts are held above **MemStart**, one for high priority and one for low priority; this allows processes at both levels to have breakpoints. Note that on bootstrapping from a link, these contexts are overwritten by the loaded code. If this is not acceptable, the values should be peeked from memory before bootstrapping from a link. The reserved area of internal memory below **MemStart** is used to implement link and event channels.

Two words of memory are reserved for timer use, **TPtrLoc0** for high priority processes and **TPtrLoc1** for low priority processes. They either indicate the relevant priority timer is not in use or point to the first process on the timer queue at that priority level.

Values of certain processor registers for the current low priority process are saved in the reserved **IntSaveLoc** locations when a high priority process pre-empts a low priority one. Other locations are reserved for extended features such as block moves.

External memory space starts at #80001000 and extends up through #00000000 to #7FFFFFFF. Memory configuration data and ROM bootstrapping code must be in the most positive address space, starting at #7FFFFF6C and #7FFFFFFE respectively. Address space immediately below this is conventionally used for ROM based code.

Two words of memory are reserved for parity checking functions, **ParityErrorAddressReg** and **ParityErrorByteReg**. These are mapped onto internal registers in the IMS T426 and may not be written to. They are updated after read accesses to external memory which contain a parity error, indicated by the **SoftParityError** or **HardParityError** signals (see section 5.1). **ParityErrorAddressReg** at address #7FFFFF68 contains the trapped word address of the parity error. The **ParityErrorByteReg** at address #7FFFFF64 contains a five bit parity error flag (see table 4.1). These registers may be read and are not cleared at power up. However, **ParityErrorByteReg** is cleared once its contents are explicitly read.

Bit	Function
0	1 = ParityError in byte 0
1	1 = ParityError in byte 1
2	1 = ParityError in byte 2
3	1 = ParityError in byte 3
4	0 = ParityError was soft, 1= ParityError was hard

Table 4.1 **ParityErrorByteReg** flags

Parity checking will not be applied to internal memory.

hi Machine map lo	Byte address		Word offsets	occam map
Reset inst	#7FFFFFFE			
Memory configuration	#7FFFFFF8 #7FFFFF6C			
ParityErrorAddressReg	#7FFFFF68		#3FFFFFDA	ParityErrorAddressReg
ParityErrorByteReg	#7FFFFF64		#3FFFFFD9	ParityErrorByteReg
	#0			
	#80001000 – Start of external memory –		#0400	
	#80000070 **MemStart**		**MemStart** #1C	
Reserved for extended functions	#8000006C #80000048			
ERegIntSaveLoc	#80000044			
STATUSIntSaveLoc	#80000040			
CRegIntSaveLoc	#8000003C			
BRegIntSaveLoc	#80000038			
ARegIntSaveLoc	#80000034			
IptrIntSaveLoc	#80000030			
WdescIntSaveLoc	#8000002C			
TPtrLoc1	#80000028	Note 1		
TPtrLoc0	#80000024			
Event	#80000020		#08	Event
Link 3 Input	#8000001C		#07	Link 3 Input
Link 2 Input	#80000018		#06	Link 2 Input
Link 1 Input	#80000014		#05	Link 1 Input
Link 0 Input	#80000010		#04	Link 0 Input
Link 3 Output	#8000000C		#03	Link 3 Output
Link 2 Output	#80000008		#02	Link 2 Output
Link 1 Output	#80000004		#01	Link 1 Output
Link 0 Output	#80000000	(Base of memory)	#00	Link 0 Output

Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 171). For details see the *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 4.1 IMS T426 memory map

5 External memory interface

The External Memory Interface (EMI) allows access to a 32 bit address space, supporting dynamic and static RAM as well as ROM and EPROM. EMI timing can be configured at **Reset** to cater for most memory types and speeds, and a program is supplied with the Transputer Development System to aid in this configuration.

There are 17 internal configurations which can be selected by a single pin (**MemConfig**) connection (page 198). If none are suitable the user can configure the interface to specific requirements, as shown in page 201.

The external memory cycle is divided into six **Tstates** with the following functions:

T1	Address setup time before address valid strobe.
T2	Address hold time after address valid strobe.
T3	Read cycle three–state or write cycle data setup.
T4	Extendable data setup time.
T5	Read or write data.
T6	Data hold.

Under normal conditions each **Tstate** may be from one to four periods **Tm** long, the duration being set during memory configuration. The default condition on **Reset** is that all **Tstates** are the maximum four periods **Tm** long to allow external initialization cycles to read slow ROM.

Period **T4** can be extended indefinitely by adding externally generated wait states. These periods are designated **W** in the following diagrams.

An external memory cycle is always an even number of periods **Tm** in length and the start of **T1** always coincides with a rising edge of **ProcClockOut**. If the total configured quantity of periods **Tm** is an odd number, one extra period **Tm** will be added at the end of **T6** to force the start of the next **T1** to coincide with a rising edge of **ProcClockOut**. This period is designated **E** in configuration diagrams (figure 5.23).

During an internal memory access cycle the external memory interface bus **MemAD2-31** reflects the word address used to access internal RAM, **MemnotWrD0** reflects the read/write operation and **MemnotRfD1** is high; all control strobes are inactive. This is true unless and until a memory refresh cycle or DMA (memory request) activity takes place, when the bus will carry the appropriate external address or data.

The bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with *peek* and *poke* (page 170). **ParityCheckEnable** will be invalid during an internal memory access.

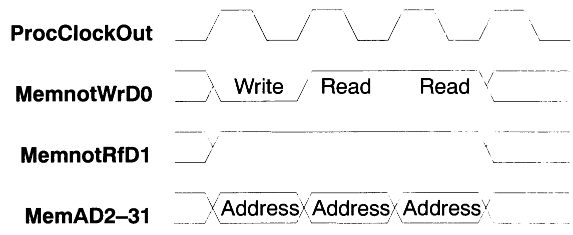


Figure 5.1 IMS T426 bus activity for internal memory cycle

5.1 Pin functions

5.1.1 MemAD2–31

External memory addresses and data are multiplexed on one bus. Only the top 30 bits of address are output on the external memory interface, using pins **MemAD2-31**. They are normally output only during **Tstates T1** and **T2**, and should be latched during this time. The data bus is 32 bits wide. It uses **MemAD2-31** for the top 30 bits and **MemnotRfD1** and **MemnotWrD0** for the lower two bits.

5.1.2 ParityDataBit0–3

Each byte of the 32 bit data bus is allocated one of four parity bits. **ParityDataBit0–3** will be generated on data output and checked (under control of **ParityCheckEnable**) on data input. **ParityDataBit0–3** is always driven during **T1** and **T2** but the data will be invalid. At all other times **ParityDataBit0–3** follows the data bus.

5.1.3 ParityCheckEnable

Parity checking can be enabled during a memory read access by setting **ParityCheckEnable** high. **ParityCheckEnable** is sampled during **T4**.

5.1.4 SoftParityError

When high this signal indicates that a parity error has occurred and that a subsequent retry cycle was successful (see section 5.6). It goes high near or after the end of the retry cycle and is reset 2 **ProcClockOut** cycles (4 **Tms**) later. The exact timing of this and **HardParityError** depends on the EMI configuration in use.

Soft errors can be monitored by connecting **SoftParityError** to event channel logic or some other external logic, enabling software to monitor both the failure rate and the likely mechanism.

5.1.5 HardParityError

When high this signal indicates that a parity error has occurred and that a subsequent retry did not pass a parity check. It follows the same general timing as **SoftParityError** but remains high until device reset.

5.1.6 ParityErrorIn, ParityErrorOut

In a network of IMS T426 devices the **ParityErrorOut** and **ParityErrorIn** signals can be daisy chained allowing the propagation of parity errors to a subsystem monitor or root node. The device that generated the error can be found by examining the internal registers.

5.1.7 notMemRd

For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Data is read by the transputer on the rising edge of this strobe, and may be removed immediately afterward. If the strobe duration is insufficient it may be extended by adding extra periods **Tm** to either or both of the **Tstates T4** and **T5**. Further extension may be obtained by inserting wait states at the end of **T4**.

5.1.8 MemnotWrD0

During **T1** and **T2** this pin will be low if the cycle is a write cycle, otherwise it will be high.

During **Tstates T3** to **T6** it becomes bit 0 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

5.1.9 notMemWrB0–3

Because the transputer uses word addressing, four write strobes are provided; one to write each byte of the word. **notMemWrB0** addresses the least significant byte.

5.1.10 notMemS0–4

To facilitate control of different types of memory and devices, the EMI is provided with five strobe outputs, four of which can be configured by the user. The strobes are conventionally assigned the functions shown in the read and write cycle diagrams, although there is no compulsion to retain these designations.

5.1.11 MemWait

Wait states can be selected by taking **MemWait** high. Externally generated wait states can be added to extend the duration of **T4** indefinitely.

5.1.12 MemnotRfD1

During **T1** and **T2**, this pin is low if the address on **MemAD2-31** is a refresh address, otherwise it is high.

During **Tstates T3** to **T6** it becomes bit 1 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

5.1.13 notMemRf

The IMS T426 can be operated with memory refresh enabled or disabled. The selection is made during memory configuration, when the refresh interval is also determined (see section 5.8).

5.1.14 RefreshPending

When high, this pin signals that a refresh cycle is pending.

5.1.15 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high. **MemGranted** follows the timing of the bus being tristated and can be used to signal to the device requesting the DMA that it has control of the bus. Note that **MemGranted** changes on the falling edge of **ProcClockOut** and can therefore be sampled to establish control of the bus on the rising edge of **ProcClockOut**.

5.1.16 MemConfig

MemConfig is an input pin used to read configuration data when setting external memory interface (EMI) characteristics (see section 5.10).

5.1.17 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn** (see section 5.2).

5.2 Processor clock

ProcClockOut is derived from the internal processor clock, which is in turn derived from **ClockIn**. Its period is equal to one internal microcode cycle time, and can be derived from the formula

TPCLPCL = TDCLDCL / PLLx

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (see chapter 10).

The time value **Tm** is used to define the duration of **Tstates** and, hence, the length of external memory cycles; its value is exactly half the period of one **ProcClockOut** cycle (0.5***TPCLPCL**), regardless of mark/space ratio of **ProcClockOut**.

Edges of the various external memory strobes coincide with rising or falling edges of **ProcClockOut**. It should be noted, however, that there is a skew associated with each coincidence. The value of skew depends on whether coincidence occurs when the **ProcClockOut** edge and strobe edge are both rising, when both are falling or if either is rising when the other is falling. Timing values given in the strobe tables show the best and worst cases. If a more accurate timing relationship is required, the exact **Tstate** timing and strobe edge to **ProcClockOut** relationships should be calculated and the correct skew factors applied from the edge skew timing table 5.2.

		T426-20		T426-25			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TPCLPCL	**ProcClockOut** period	48	52	38	42	ns	2
TPCHPCL	**ProcClockOut** pulse width high	13.5	28.5	8.5	23.5	ns	2
TPCLPCH	**ProcClockOut** pulse width low	**a**		**a**		ns	2, 3, 4
Tm	**ProcClockOut** half cycle	24	26	19	21	ns	2
TPCstab	**ProcClockOut** stability		8		8	%	1,2

Notes

1 Stability is the variation of cycle periods between two consecutive cycles, measured at corresponding points on the cycles.

2 This parameter is sampled and not 100% tested.

3 **a** is TPCLPCL – TPCHPCL.

4 This is a nominal value.

Table 5.1 ProcClockOut timing

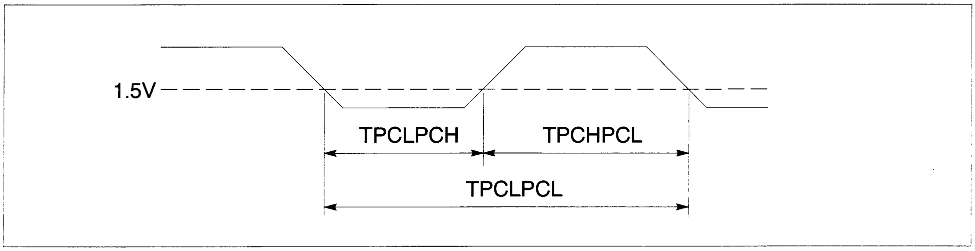


Figure 5.2 IMS T426 **ProcClockOut** timing

5.3 Strobes

notMemS0 is a fixed format strobe. Its leading edge is always coincident with the start of **T2** and its trailing edge always coincident with the end of **T5**.

The leading edge of **notMemS1** is always coincident with the start of **T2**, but its duration may be configured to be from zero to 31 periods **Tm**. Regardless of the configured duration, the strobe will terminate no later than the end of **T6**. The strobe is sometimes programmed to extend beyond the normal end of the complete cycle **Tmx**. When wait states are inserted into an EMI cycle the end of **Tmx** is delayed, but the potential active duration of the strobe is not altered. Thus the strobe can be configured to terminate relatively early under certain conditions (page 190). If **notMemS1** is configured to be zero it will never go low.

notMemS2, **notMemS3** and **notMemS4** are identical in operation. They all terminate at the end of **T5**, but the start of each can be delayed from one to 31 periods **Tm** beyond the start of **T2**. If the duration of one of these strobes would take it past the end of **T5** it will stay high. This can be used to cause a strobe to become active only when wait states are inserted. If one of these strobes is configured to zero it will stay low for **T1** to **T5** and go high for the initial **Tm** of **T6** and then go low again. Figure 5.3 shows the effect of **Wait** on strobes in more detail; each division on the scale is one period **Tm**.

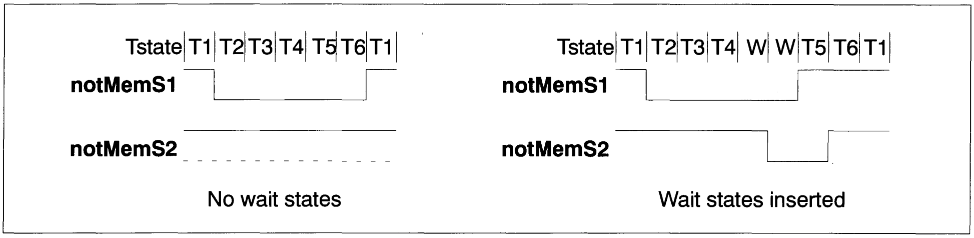


Figure 5.3 IMS T426 effect of wait states on strobes

Symbol	Parameter	T426-20		T426-25		Units	Note
		Min	Max	Min	Max		
TPCHSOH	**notMemS0** rising from **ProcClockOut** rising	–6	4	–6	4	ns	1
TPCLSOH	**notMemS0** rising from **ProcClockOut** falling	–5	10	–5	10	ns	1
TPCHSOL	**notMemS0** falling from **ProcClockOut** rising	–8	3	–8	3	ns	1
TPCLSOL	**notMemS0** falling from **ProcClockOut** falling	–5	7	–5	7	ns	1

Notes

1 Sampled, not 100% tested.

Table 5.2 **notMemS0** to **ProcClockOut** skew

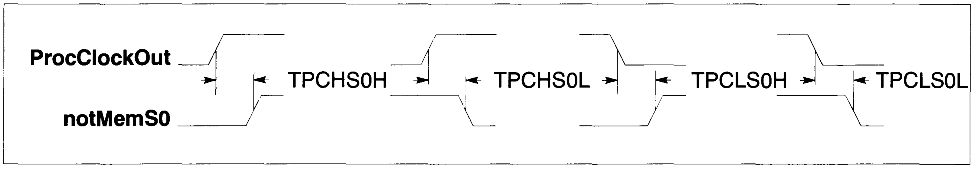


Figure 5.4 IMS T426 skew of **notMemS0** to **ProcClockOut**

The following table should be read in conjunction with read cycle diagrams 5.5 and 5.6 and write cycle diagram 5.7.

Symbol	ⓝ	Parameter	T426-20		T426-25		Units	Notes
			Min	Max	Min	Max		
TaVS0L		Address setup before **notMemS0**	a–8		a–8		ns	1
TS0LaX		Address hold after **notMemS0**	b–8	b+8	b–8	b+8	ns	2
TS0LS0H		**notMemS0** pulse width low	c–5	c+6	c–5	c+6	ns	3
TS0LS1L	1	**notMemS1** from **notMemS0**	–4	4	–4	4	ns	10
TS0LS1H	5	**notMemS1** end from **notMemS0**	d–1	d+9	d–1	d+9	ns	4, 6
TS0HS1H	9	**notMemS1** end from **notMemS0** end	e–8	e+4	e–8	e+4	ns	5, 6
TS0LS2L	2	**notMemS2** delayed after **notMemS0**	f–6	f+5	f–6	f+5	ns	7
TS0LS2H	6	**notMemS2** end from **notMemS0**	c–5	c+7	c–5	c+7	ns	3,10
TS0HS2H	10	**notMemS2** end from **notMemS0** end	–4	7	–4	7	ns	10
TS0LS3L	3	**notMemS3** delayed after **notMemS0**	f–6	f+5	f–6	f+5	ns	7
TS0LS3H	7	**notMemS3** end from **notMemS0**	c–5	c+7	c–5	c+7	ns	3,10
TS0HS3H	11	**notMemS3** end from **notMemS0** end	–4	7	–4	7	ns	10
TS0LS4L	4	**notMemS4** delayed after **notMemS0**	f–6	f+5	f–6	f+5	ns	7
TS0LS4H	8	**notMemS4** end from **notMemS0**	c–5	c+7	c–5	c+7	ns	3,10
TS0HS4H	12	**notMemS4** end from **notMemS0** end	–4	7	–4	7	ns	10
Tmx		Complete external memory cycle						8,9

Notes

1 **a** is **T1** where **T1** can be from one to four periods **Tm** in length.

2 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

3 **c** is total of **T2**+**T3**+**T4**+**Twait**+**T5** where **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

4 **d** can be from zero to 31 periods **Tm** in length.

5 **e** can be from –27 to +4 periods **Tm** in length.

6 If the configuration would cause the strobe to remain active past the end of **T6** it will go high at the end of **T6**. If the strobe is configured to zero periods **Tm** it will remain high throughout the complete cycle **Tmx**.

7 **f** can be from zero to 31 periods **Tm** in length. If this length would cause the strobe to remain active past the end of **T5** it will go high at the end of **T5**. If the strobe value is zero periods **Tm** it will remain low throughout **T1** to **T5** and go high for the initial **Tm** of **T6**.

8 **Tmx** is one complete external memory cycle comprising the total of **T1**+**T2**+**T3**+**T4**+**Twait**+**T5**+**T6** where **T1**, **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length, **T6** can be from one to five periods **Tm** in length and **Twait** may be zero or any number of periods **Tm** in length.

9 Guaranteed, but not tested.

10 Sampled, not 100% tested.

Table 5.3 IMS T426 strobe timing

5.4 Read cycle

Byte addressing is carried out internally by the transputer for read cycles. For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Read cycle data may be set up on the data bus at any time after the start of **T3**, but must be valid when the transputer reads it at the end of **T5**. Data may be removed any time during **T6**, but must be off the bus no later than the end of that period.

Each byte of the 32 bit data bus is checked for ***odd*** parity. The result of this parity calculation will be ignored unless **ParityCheckEnable** was held high during **T4**, in which case the result will be compared to the relevant **ParityDataBit0–3** and appropriate action taken (see section 5.6).

Due to the overheads of parity checking, two extra **Tm** periods may be added to the back end of the memory cycle. This will depend upon the memory configuration in use and never happens for cycles where **ParityCheckEnable** is held low or if the total number of **Tm**'s in **T6** (including the **E** period) is 3, 4 or 5. These extra **Tm**'s are designated **P** in the timing diagrams and always occur between the end of **T6** and the beginning of **T1**. The relative timing of the strobes is unaffected.

In the read cycle timing diagrams **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**.

Symbol	Parameter	T426–20			T426–25			Units	Notes
		Min	Nom	Max	Min	Nom	Max		
TaZdV	Address tristate to data valid	0			0			ns	3
TdVRdH	Data setup before read	25			20			ns	
TRdHdX	Data hold after read	0			0			ns	3
TS0LRdL	**notMemS0** before start of read	**a**–4	**a**	**a**+4	**a**–4	**a**	**a**+4	ns	1
TS0HRdH	End of read from end of **notMemS0**	–4		4	–4		4	ns	
TRdLRdH	Read period	**b**–3		**b**+5	**b**–3		**b**+5	ns	2
TS0LPEH	**ParityCheckEnable** set up	**a**–25			**a**–20			ns	4
TS0LPEL	**ParityCheckEnable** hold	**a**+50			**a**+40			ns	4

Notes

1 **a** is total of **T2**+**T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

2 **b** is total of **T4**+**Twait**+**T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

3 Guaranteed, but not tested.

4 Provisional timing.

Table 5.4 Read cycle timing

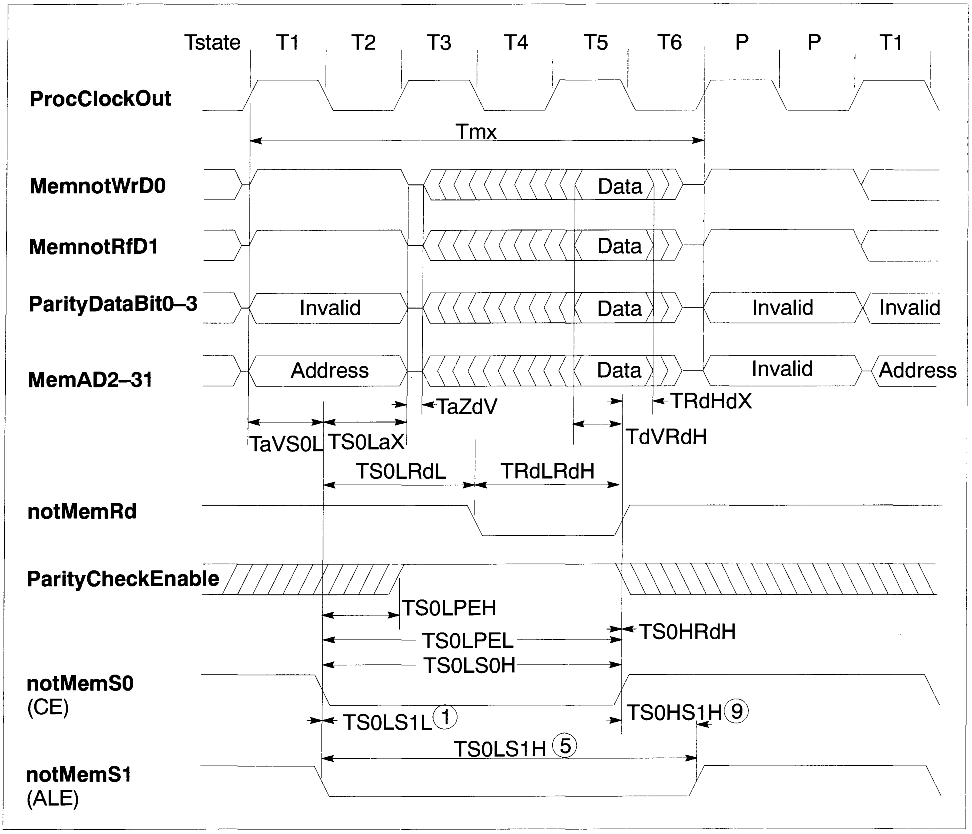


Figure 5.5 IMS T426 external read cycle: static memory with parity checking enabled

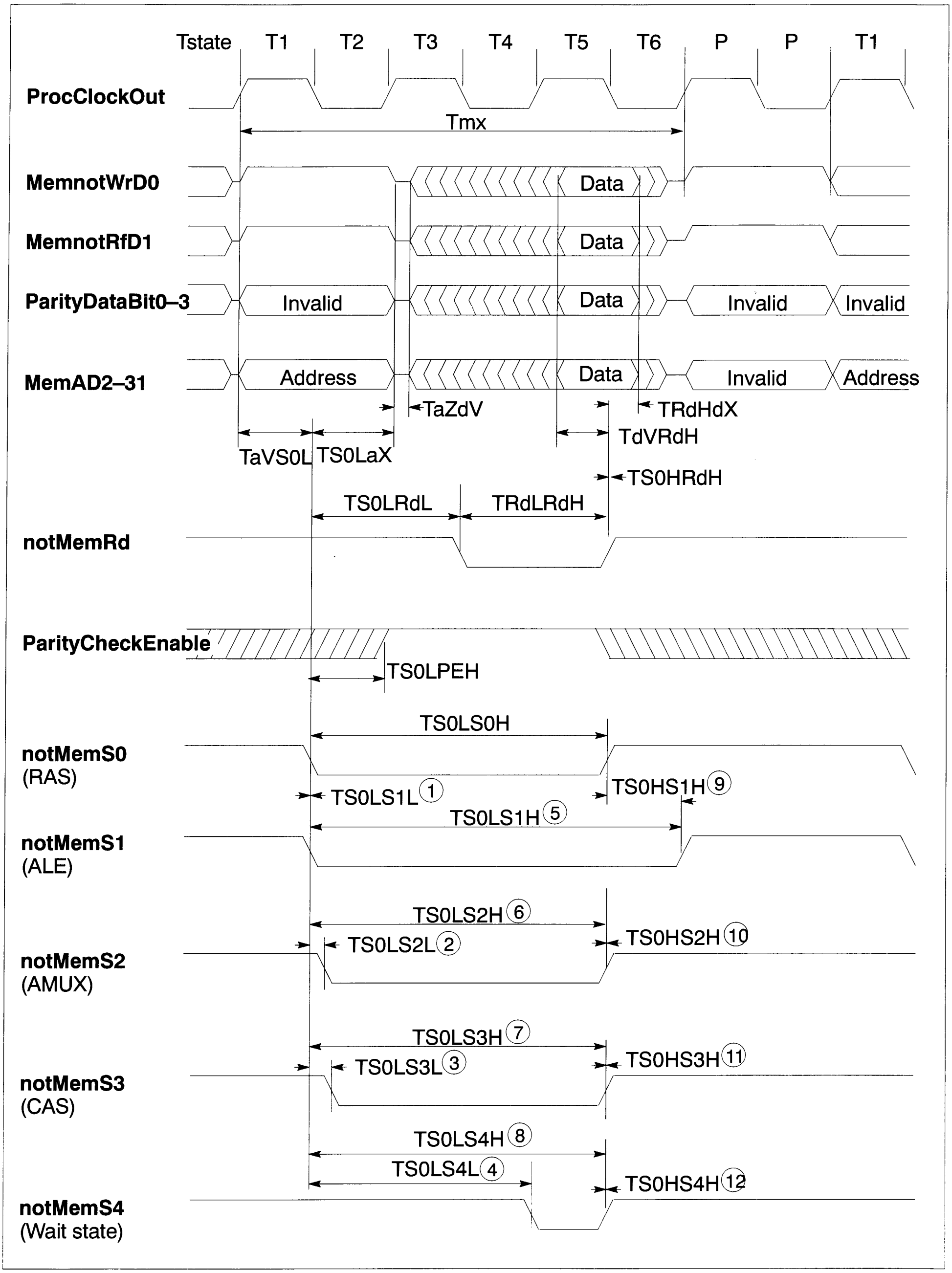


Figure 5.6 IMS T426 external read cycle: dynamic memory with parity checking enabled

5.5 Write cycle

For write cycles the relevant bytes in memory are addressed by the write strobes **notMemWrB0-3**. If a particular byte is not to be written, then the corresponding data outputs are tristated. The **ParityDataBit0–3** are generated regardless of **ParityCheckEnable** and are tristated or driven together with their respective bytes. **ParityCheckEnable** is ignored during write cycles.

For a write cycle, pin **MemnotWrD0** will be low during **T1** and **T2**. Parity data is placed on the bus at the start of **T3**, as is write data. Data is removed at the end of **T6**. If **T6** is extended to force the next cycle **Tmx** (page 178) to start on a rising edge of **ProcClockOut**, data will be valid during this time also.

The transputer has both early and late write cycle modes. For a late write cycle the relevant write strobes **notMemWrB0-3** are low during **T4** and **T5**; for an early write they are also low during **T3**. Data should be latched into memory on the rising edge of the strobes in both cases, although it is valid until the end of **T6**. If the strobe duration is insufficient, it may be extended at configuration time by adding extra periods **Tm** to either or both of **Tstates T4** and **T5** for both early and late modes. For an early cycle they may also be added to **T3**. Further extension may be obtained by inserting wait states at the end of **T4**. If the data hold time is insufficient, extra periods **Tm** may be added to **T6** to extend it.

In the write cycle timing diagram **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**. The strobe is inactive during internal memory cycles.

Symbol	Parameter	T426-20		T426-25		Units	Notes
		Min	Max	Min	Max		
TdVWrH	Data setup before write	d–7	d+10	d–7	d+10	ns	1, 5
TWrHdX	Data hold after write	a–10	a+5	a–10	a+5	ns	1, 2
TS0LWrL	**notMemS0** before start of early write	b–5	b+5	b–5	b+5	ns	1, 3
	notMemS0 before start of late write	c–5	c+5	c–5	c+5	ns	1, 4
TS0HWrH	End of write from end of **notMemS0**	–5	4	–5	4	ns	1,7
TWrLWrH	Early write pulse width	d–4	d+7	d–4	d+7	ns	1, 5
	Late write pulse width	e–4	e+7	e–4	e+7	ns	1, 6

Notes

1 Timing is for all write strobes **notMemWrB0-3**.

2 **a** is **T6** where **T6** can be from one to five periods **Tm** in length.

3 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

4 **c** is total of **T2**+**T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

5 **d** is total of **T3**+**T4**+**Twait**+**T5** where **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

6 **e** is total of **T4**+**Twait**+**T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

7 Sampled, not 100% tested.

Table 5.5 Write cycle timing

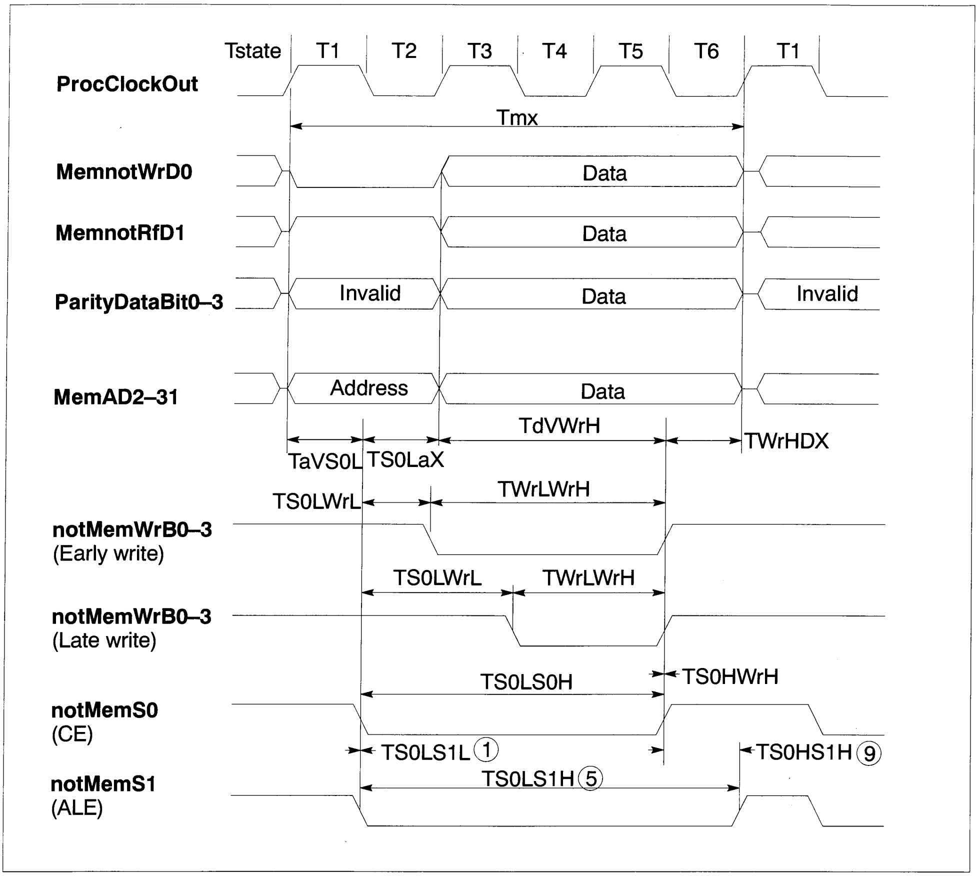


Figure 5.7 IMS T426 external write cycle

5.6 Parity errors

When **ParityCheckEnable** is held high during read cycles, all data read by the IMS T426 is checked against *true parity*. If no error is found data is transmitted to the requestor (links or CPU) as normal. However, if an error occurs the following will happen;

- The parity error flags (one flag per byte) are set in the memory mapped register **ParityErrorReg**.
- The address of the errant read access is stored in the memory mapped register **ParityErrorAddressReg**.
- Data accessed during the cycle is discarded.

The memory interface then repeats the current access, once again issuing the same address. If the retry also fails parity checking, the following will occur;

- **HardParityError** and **ParityErrorOut** go high during the second low **Tm** of **ProcClockOut** after **T5** (see figure 5.8).
- The errant access is not allowed to complete, leaving the requestor starved of data and the memory bus stalled. Refresh cycles continue.

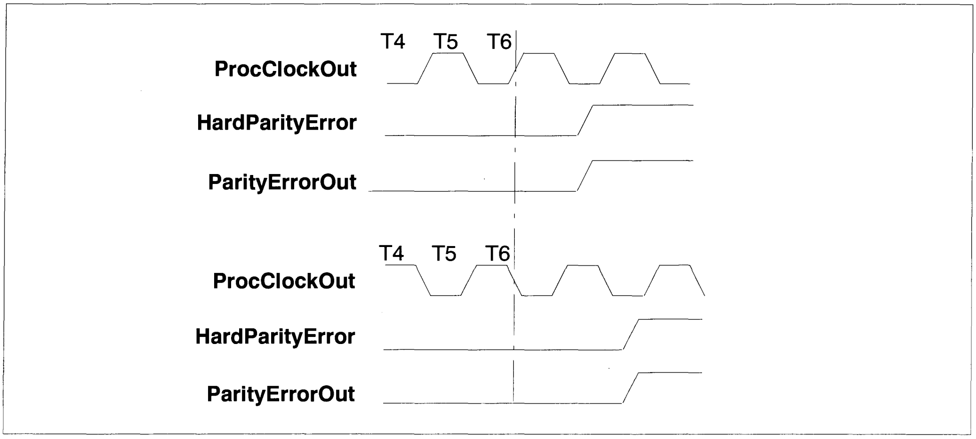


Figure 5.8 **HardParityError** and **ParityErrorOut** relative to **T5**.

If the retry is successful the machine will continue to run normally, but **SoftParityError** will be asserted. **SoftParityError** follows the general timing of **HardParityErrcr**, but is reset four **Tm** periods (two cycles of **ProcClockOut**) later.

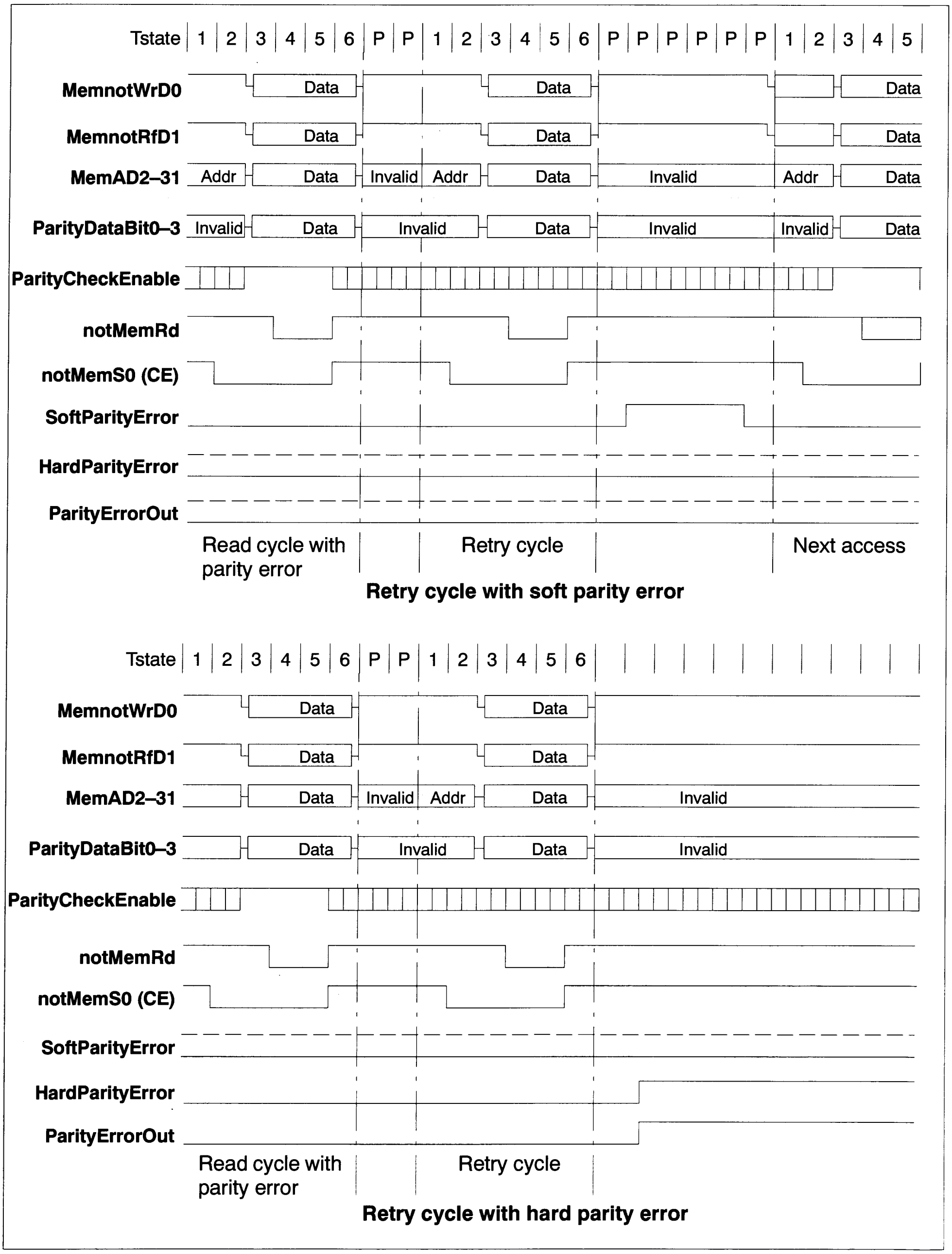
Once a hard parity error has occurred, one of the following actions should be taken;

- Reset the IMS T426 and reboot.
- Apply **Analyse** and **Reset**.

Resetting the IMS T426 re-configures the memory interface without destroying the contents of registers **ParityErrorReg** and **ParityErrorAddressReg**.

Applying **Analyse** and **Reset** releases the memory bus, leaving the IMS T426 in the normal analyse state. The contents of the **ParityErrorReg** and **ParityErrorAddressReg** can then be read by software.

Figure 5.9 illustrates the retry mechanism when soft and hard parity errors are found respectively; each division on the scale is one period **Tm**. Note that, in the case of a successful retry, an extra four **Tm** periods (marked **P** in the diagram) will be added after the retry cycle. The number is dependent upon configuration and will be four in all cases except when the number of **Tm**'s in **T6** is five (four plus an **E** period), when only two **Tm**'s will be inserted. These will be in addition to any **E** or **P** periods currently existing due to the configuration in use.



Retry cycle with soft parity error

Retry cycle with hard parity error

Figure 5.9 Retry cycles for soft and hard parity errors

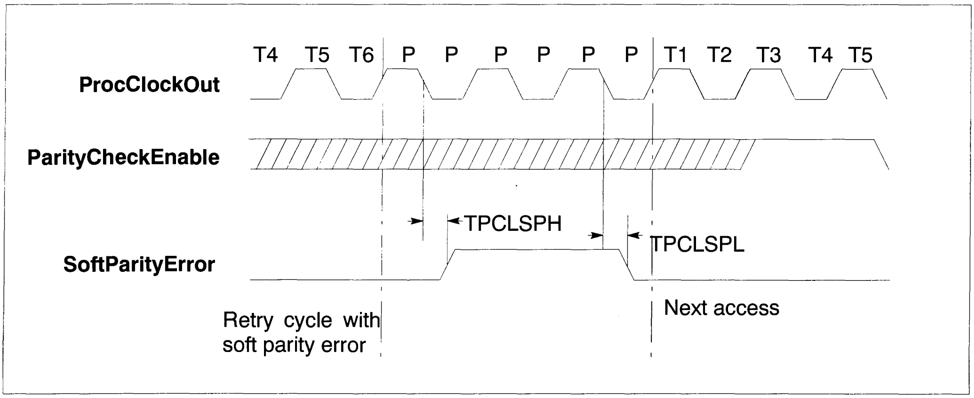


Figure 5.10 Soft parity error timing

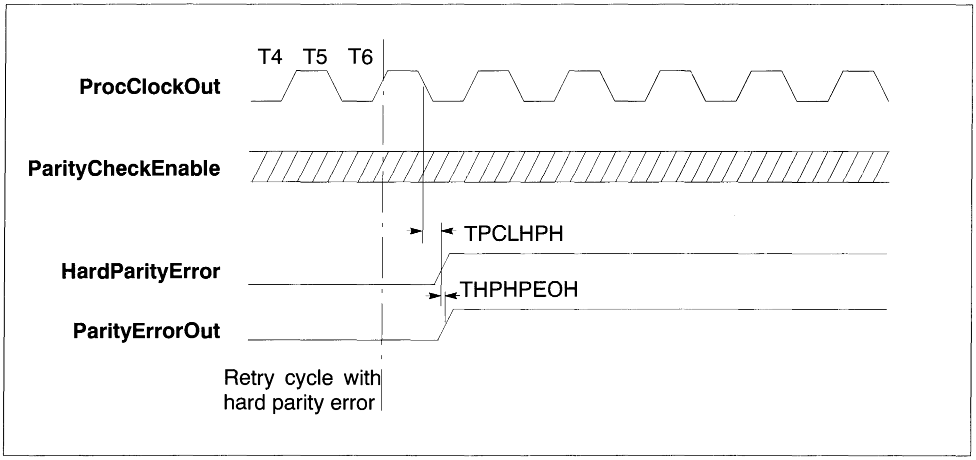


Figure 5.11 Hard parity error timing

Symbol	Parameter	T426-20		T426-25		Units	Note
		Min	Max	Min	Max		
TPCLSPH	**SoftParityError** high after **ProcClockOut** low	8	17	8	17	ns	1
TPCLSPL	**SoftParityError** low after **ProcClockOut** low	8	17	8	17	ns	1
TPCLHPH	**HardParityError** high after **ProcClockOut** low	8	17	8	17	ns	1
THPHPEOH	**ParityErrorOut** lag behind **HardParityError**	0.5	4.5	0.5	4.5	ns	1

Notes

1 **ProcClockOut**, **SoftParityError**, **HardParityError** and **ParityErrorOut** are assumed to be loaded with 50 pF.

Table 5.6 Parity error timing

5.7 Wait

Taking **MemWait** high with the timing shown (figure 5.12) will extend the duration of **T4**. **MemWait** is sampled close to the falling edge of **ProcClockOut** prior to, but not at, the end of **T4**. By convention, **notMemS4** is used to synchronize wait state insertion. If this or another strobe is used, its delay should be such as to take the strobe low an even number of periods **Tm** after the start of **T1**, to coincide with a rising edge of **ProcClockOut**.

MemWait may be kept high indefinitely, although if dynamic memory refresh is used it should not be kept high long enough to interfere with refresh timing. **MemWait** operates normally during all cycles, including refresh and configuration cycles. It does not affect internal memory access in any way.

If the start of **T5** would coincide with a falling edge of **ProcClockOut** an extra wait period **Tm** (**EW**) is generated by the EMI to force coincidence with a rising edge. Rising edge coincidence is only forced if wait states are added, otherwise coincidence with a falling edge is permitted.

Symbol	Parameter	T426–20		T426–25		Units	Notes
		Min	Max	Min	Max		
TPCLWtH	Wait setup	10		10		ns	1, 2
TPCLWtL	Wait hold	8		8		ns	1, 2
TWtLWtH	Delay before re-assertion of Wait	50		40			3

Notes

1 **ProcClockOut** load should not exceed 50pF.

2 If wait period exceeds refresh interval, refresh cycles will be lost.

3 Guaranteed, but not tested.

Table 5.7 Memory wait timing

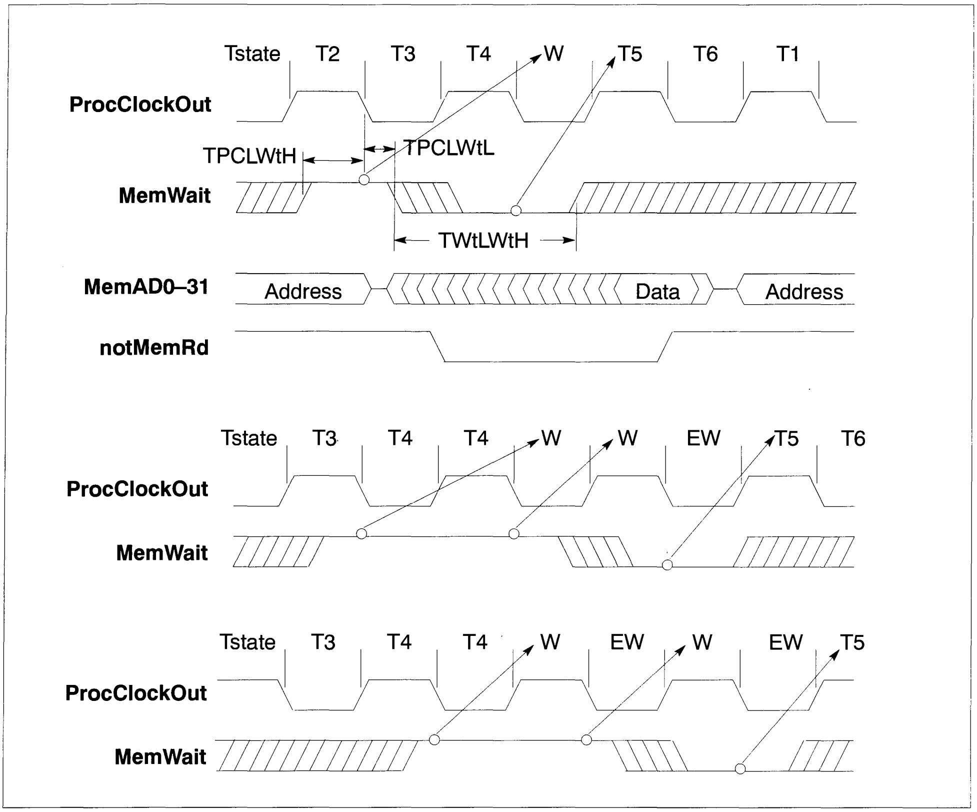


Figure 5.12 IMS T426 memory wait timing

5.8 Memory refresh

The **RefreshPending** pin is asserted high when the external memory interface is about to perform a refresh cycle. It remains high until the refresh cycle is started by the transputer. The minimum time for the **RefreshPending** pin to be high is for one cycle of **ProcClockOut** (two periods **Tm**), when the EMI was not about to perform a memory read or write. If the EMI was held in the tristate condition with **MemGranted** asserted, then **RefreshPending** will be asserted when the refresh controller in the EMI is ready to perform a refresh. **MemReq** may be re-asserted any time after the commencement of the refresh cycle. **RefreshPending** changes state near the rising edge of **ProcClockOut** and can therefore be sampled by the falling edge of **ProcClockOut**.

If no DMA is active then refresh will be performed following the end of the current internal or external memory cycle. If DMA is active the transputer will wait for DMA to terminate before commencing the refresh cycle. Unlike **MemnotRfD1**, **RefreshPending** is never tristated and can thus be interrogated by the DMA device; the DMA cycle can then be suspended, at the discretion of the DMA device, to allow refresh to take place.

The simple circuit of figure 5.13 will suspend DMA requests from the external logic when **RefreshPending** is asserted, so that a memory refresh cycle can be performed. DMA is restored on completion of the refresh cycle. The transputer will not perform an external memory cycle other than a refresh cycle, using this method, until the requesting device removes its DMA request.

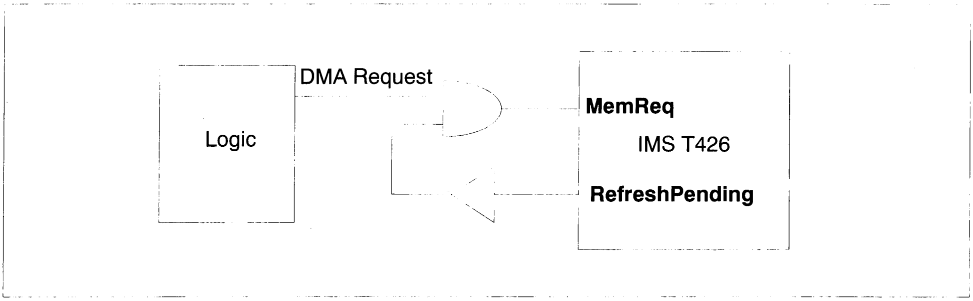


Figure 5.13 IMS T426 refresh with DMA

When refresh is disabled no refresh cycles occur. During the post-**Reset** period eight dummy refresh cycles will occur with the appropriate timing but with no bus or strobe activity.

A refresh cycle uses the same basic external memory timing as a normal external memory cycle, except that it starts two periods **Tm** before the start of **T1**. If a refresh cycle is due during an external memory access, it will be delayed until the end of that external cycle. Two extra periods **Tm** (periods **R** in diagrams 5.14 and 5.15) will then be inserted between the end of **T6** of the external memory cycle and the start of **T1** of the refresh cycle itself. The refresh address and various external strobes become active approximately one period **Tm** before **T1**. Bus signals are active until the end of **T2**, whilst **notMemRf** remains active until the end of **T6**.

Note that refresh cycles are not held up by **P** periods existing as part of a configuration or as a result of a parity error. The timing of refresh cycles relative to the end of **T6** of the previous cycles is unaffected by parity errors.

For a refresh cycle, **MemnotRfD1** goes low when **notMemRf** goes low and **MemnotWrD0** goes high with the same timing as **MemnotRfD1**. All the address lines share the same timing, but only **MemAD2-11** give the refresh address. **MemAD12-30** stay high during the address period, whilst **MemAD31** remains low. Refresh cycles generate strobes **notMemS0-4** with timing as for a normal external cycle, but **notMemRd** and **notMemWrB0-3** remain high. **MemWait** operates normally during refresh cycles.

Refresh cycles do not interrupt internal memory accesses, although the internal addresses cannot be reflected on the external bus during refresh.

Symbol	Parameter	T426-20			T426-20			Units	Notes
		Min	Nom	Max	Min	Nom	Max		
TRfLRfH	Refresh pulse width low	**a**–2		**a**+9	**a**–2		**a**+9	ns	1
TRaVS0L	Refresh address setup before **notMemS0**	**b**–12			**b**–12			ns	
TRfLS0L	Refresh indicator setup before **notMemS0**	**b**–4	**b**	**b**+6	**b**–4	**b**	**b**+6	ns	2
TRPHPCL	**RefreshPending** setup	0	12	30	0	12	30	ns	
TPCLRPL	**RefreshPending** hold	0	28	55	0	28	55	ns	

Notes

1 **a** is total **Tmx**+**Tm**.

2 **b** is total **T1**+**Tm** where **T1** can be from one to four periods **Tm** in length.

Table 5.8 Memory refresh

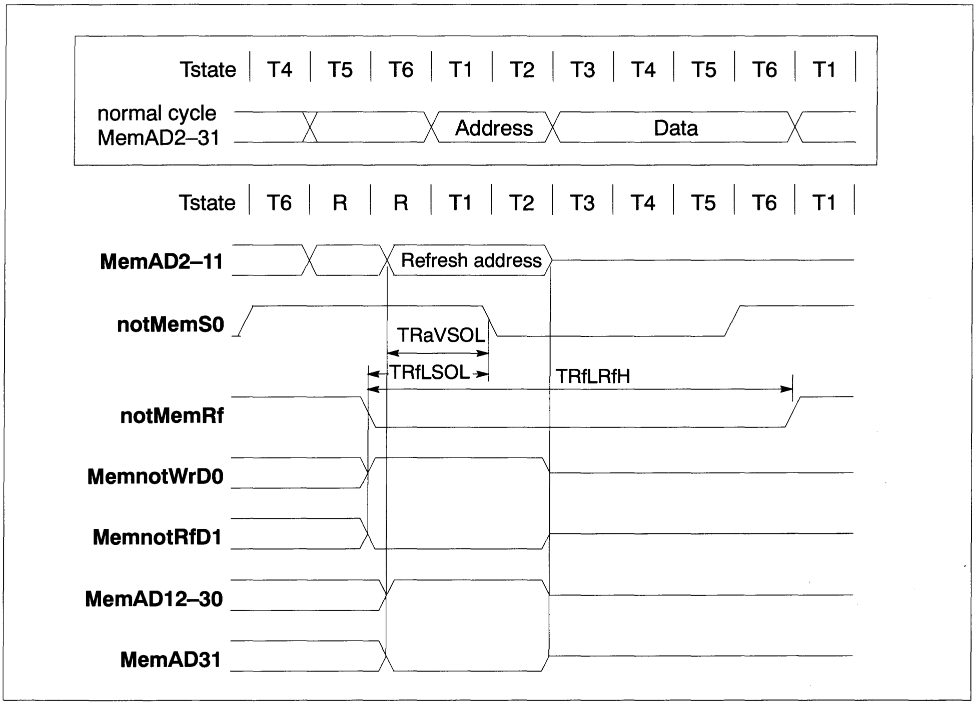


Figure 5.14 IMS T426 refresh cycle timing

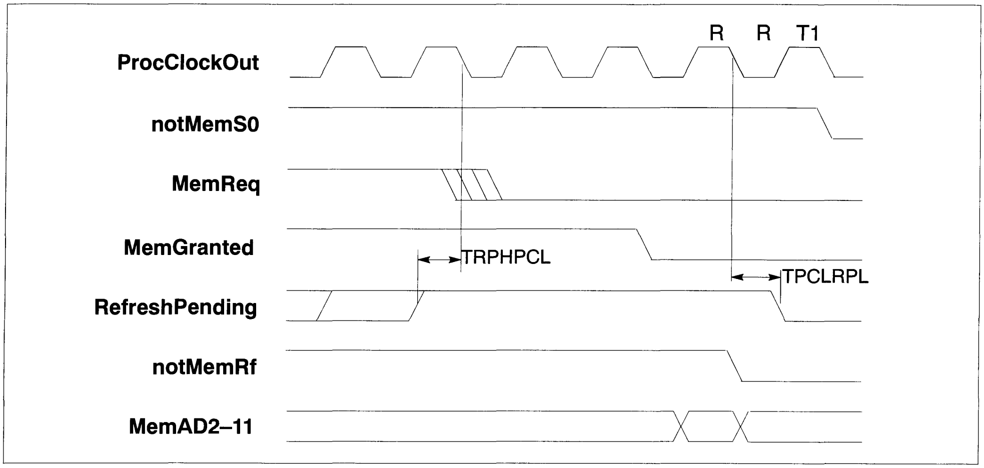


Figure 5.15 IMS T426 **RefreshPending** timing diagram

5.9 Direct memory access

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. The transputer samples **MemReq** just before falling edges **ProcClockOut**. To guarantee taking over the bus immediately following either a refresh or external memory cycle, **MemReq** must be sampled at least four periods **Tm** before the end of **T6**. In the absence of an external memory cycle, the address bus is tristated two periods **Tm** after the **ProcClockOut** rising edge which follows the sample.

Removal of **MemReq** is sampled just before falling edges of **ProcClockOut** and **MemGranted** is removed synchronously with the next falling edge of **ProcClockOut** which follows the sample. If accurate timing of DMA is required, the setup time relative to **ProcClockOut** must be met. Further external bus activity, either refresh, external cycles or reflection of internal cycles, will commence at the next rising edge of **ProcClockOut**.

The strobes (**notMemS0–4** and **notMemWrB0–4**) are left in their inactive states during DMA. DMA cannot interrupt a refresh or external memory cycle, and outstanding refresh cycles will occur before the bus is released to DMA. DMA does not interfere with internal memory cycles in any way, although a program running in internal memory would have to wait for the end of DMA before accessing external memory. DMA cannot access internal memory. If DMA extends longer than one refresh interval (Memory refresh configuration coding, table 5.12), the DMA user becomes responsible for refresh (see section 5.8). DMA may also inhibit an internally running program from accessing external memory.

DMA allows a bootstrap program to be loaded into external RAM ready for execution after reset. If **MemReq** is held high throughout reset, **MemGranted** will be asserted before the bootstrap sequence begins. **MemReq** must be high at least one period **TDCLDCL** of **ClockIn** before **Reset**. The circuit should be designed to ensure correct operation if **Reset** could interrupt a normal DMA cycle.

		T426–20		T426–25			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Note**
TMRHPCL	**MemReq** setup before **ProcClockOut** falling	3	14	3	14	ns	1
TPCLMGH	**MemReq** response time	96	110	77	89	ns	2
TMRLPCL	**Memreq** removal before **ProcClockOut** falling	4	16	4	15		
TPCLMGL	**MemReq** end response time	50	66	40	54	ns	
TADZMGH	Bus tristate before **MemGranted**	0	27	0	22	ns	
TMGLADV	Bus active after end of **MemGranted**	0	32	0	26	ns	

Notes

1 Setup time need only be met to guarantee sampling on this edge.

2 If an external cycle is active, maximum time could be (1 EMI cycle **Tmx**)+(1 refresh cycle **TRfLRfH**)+(6 periods **Tm**).

Table 5.9 Memory request timing

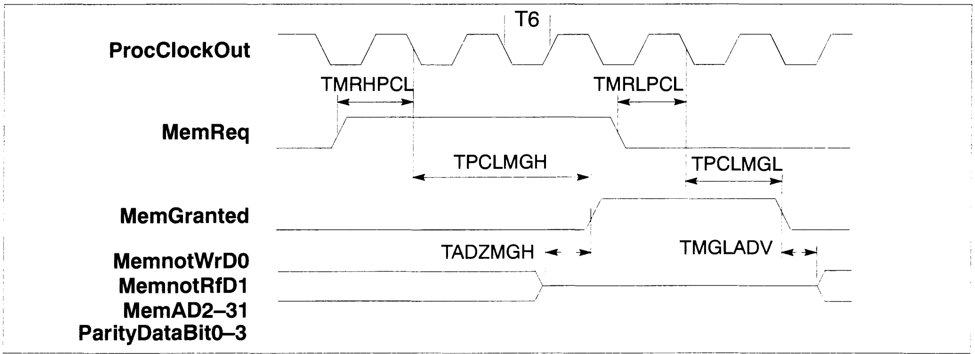
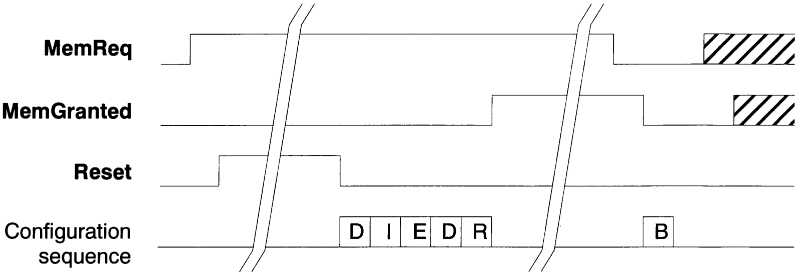


Figure 5.16 IMS T426 memory request timing



D Pre– and post–configuration delays (figure 3.3)
I Internal configuration sequence
E External configuration sequence
R Initial refresh sequence
B Bootstrap sequence

Figure 5.17 IMS T426 DMA sequence at reset

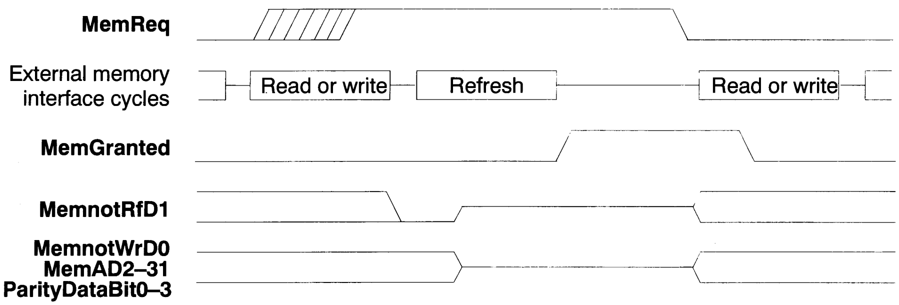


Figure 5.18 IMS T426 operation of **MemReq**, **MemGranted** with external, refresh memory cycles

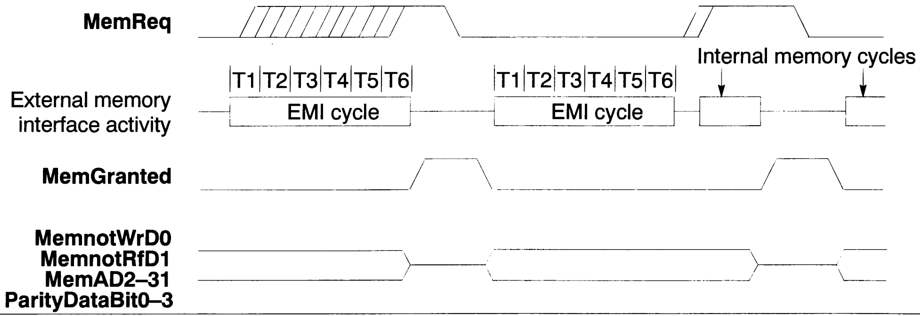


Figure 5.19 IMS T426 operation of **MemReq**, **MemGranted** with external, internal memory cycles

5.10 Memory configuration

MemConfig is an input pin used to read configuration data when setting EMI characteristics. It is read by the processor on two occasions after **Reset** goes low; first to check if one of the preset internal configurations is required, then to determine a possible external configuration.

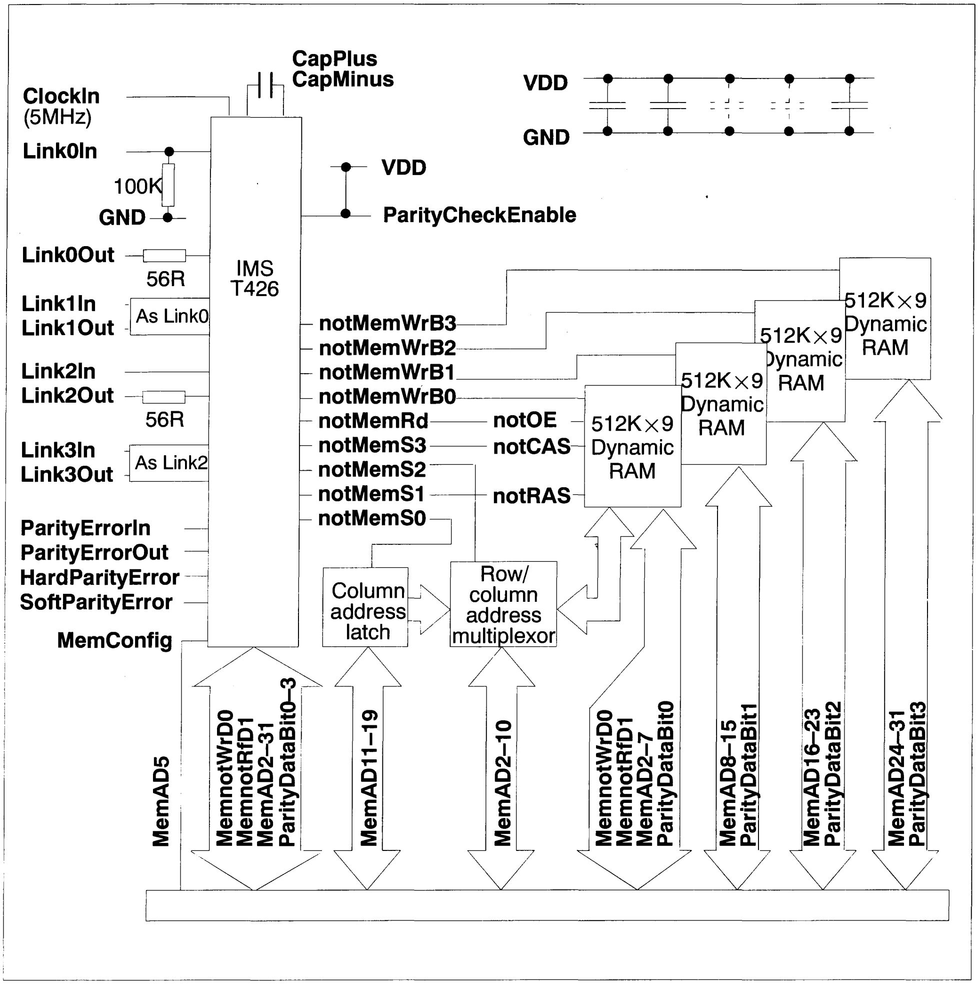


Figure 5.20 IMS T426 dynamic RAM application

5.10.1 Internal configuration

The internal configuration scan comprises 64 periods **TDCLDCL** of **ClockIn** during the internal scan period of 144 **ClockIn** periods. **MemnotWrD0**, **MemnotRfD1** and **MemAD2-32** are all high at the beginning of the scan. Starting with **MemnotWrD0**, each of these lines goes low successively at intervals of two **ClockIn** periods and stays low until the end of the scan. If one of these lines is connected to **MemConfig** the preset internal configuration mode associated with that line will be used as the EMI configuration. The default configuration is that defined in the table for **MemAD31**; connecting **MemConfig** to **VDD** will also produce this default configuration. Note that only 17 of the possible configurations are valid, all others remain at the default configuration.

	Duration of each Tstate periods Tm						Strobe coefficient				Write cycle	Refresh interval	Cycle time
Pin	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**s1**	**s2**	**s3**	**s4**	**type**	**ClockIn cycles**	**Proc cycles**
MemnotWrD0	1	1	1	1	1	1	30	1	3	5	late	72	3
MemnotRfD1	1	2	1	1	1	2	30	1	2	7	late	72	4
MemAD2	1	2	1	1	2	3	30	1	2	7	late	72	5
MemAD3	2	3	1	1	2	3	30	1	3	8	late	72	6
MemAD4	1	1	1	1	1	1	3	1	2	3	early	72	3
MemAD5	1	1	2	1	2	1	5	1	2	3	early	72	4
MemAD6	2	1	2	1	3	1	6	1	2	3	early	72	5
MemAD7	2	2	2	1	3	2	7	1	3	4	early	72	6
MemAD8	1	1	1	1	1	1	30	1	2	3	early	†	3
MemAD9	1	1	2	1	2	1	30	2	5	9	early	†	4
MemAD10	2	2	2	2	4	2	30	2	3	8	late	72	7
MemAD11	3	3	3	3	3	3	30	2	4	13	late	72	9
MemAD12	1	1	2	1	2	1	4	1	2	3	early	72	4
MemAD13	2	1	2	1	2	2	5	1	2	3	early	72	5
MemAD14	2	2	2	1	3	2	6	1	3	4	early	72	6
MemAD15	2	1	2	3	3	3	8	1	2	3	early	72	7
MemAD31	4	4	4	4	4	4	31	30	30	18	late	72	12

† Provided for static RAM only.

Table 5.10 IMS T426 internal configuration coding

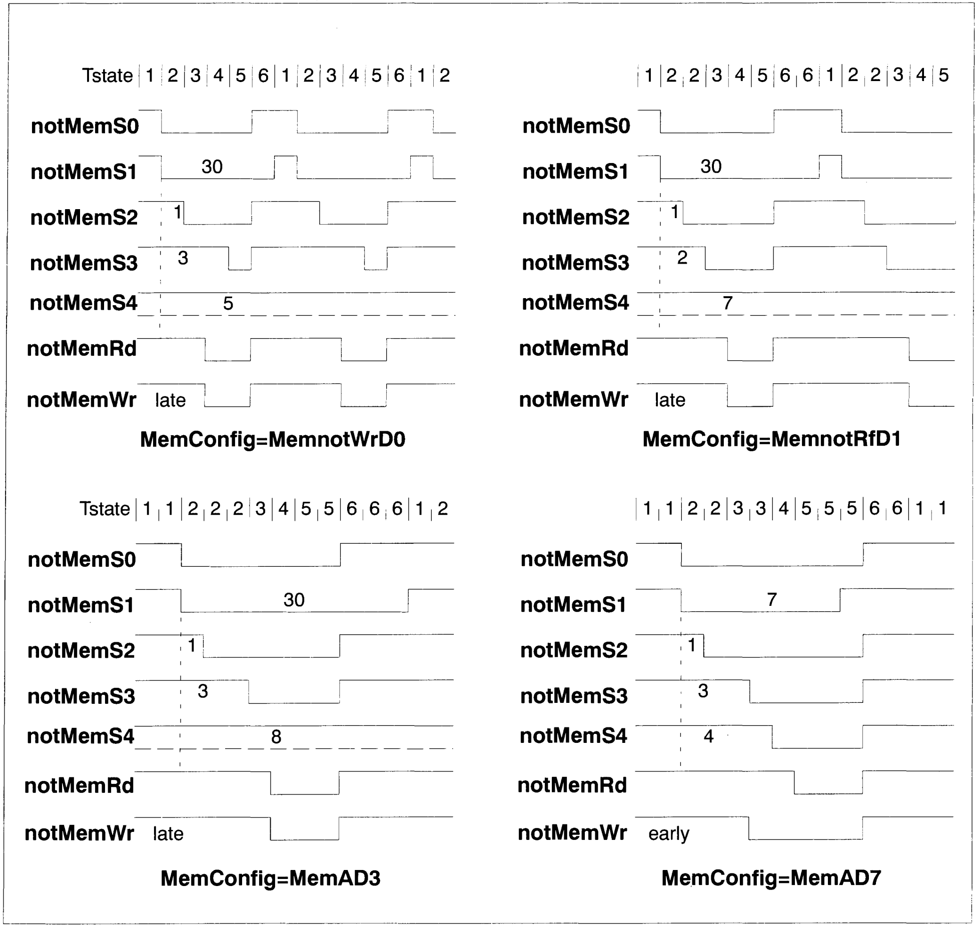
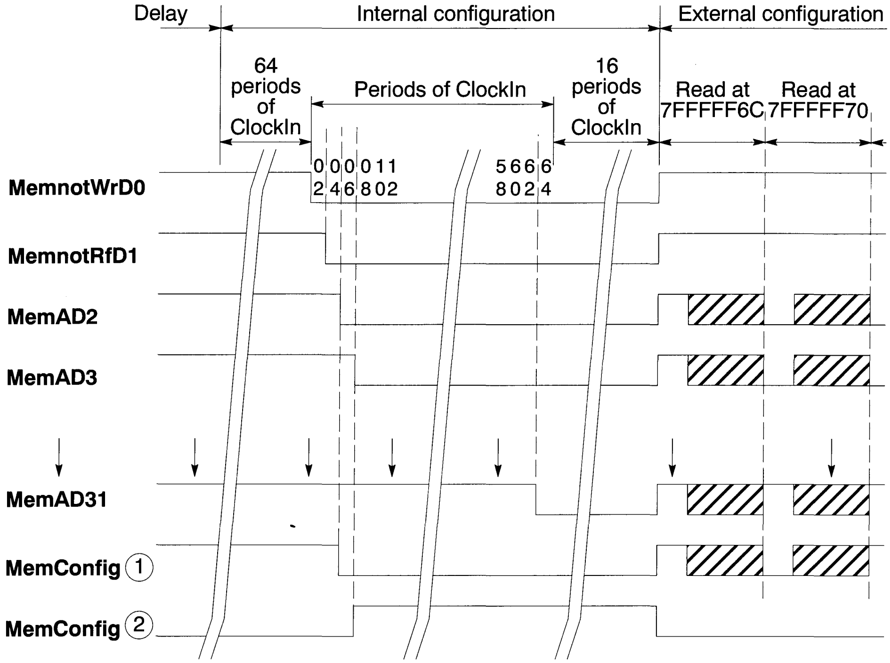


Figure 5.21 IMS T426 internal configuration



① Internal configuration: **MemConfig** connected to **MemAD2**

② External configuration: **MemConfig** connected to inverse of **MemAD3**

Figure 5.22 IMS T426 internal configuration scan

5.10.2 External configuration

If **MemConfig** is held low until **MemnotWrD0** goes low the internal configuration is ignored and an external configuration will be loaded instead. An external configuration scan always follows an internal one, but if an internal configuration occurs any external configuration is ignored.

The external configuration scan comprises 36 successive external read cycles, using the default EMI configuration preset by **MemAD31**. However, instead of data being read on the data bus as for a normal read cycle, only a single bit of data is read on **MemConfig** at each cycle. Addresses put out on the bus for each read cycle are shown in table 5.11, and are designed to address ROM at the top of the memory map. The table shows the data to be held in ROM; data required at the **MemConfig** pin is the inverse of this.

MemConfig is typically connected via an inverter to **MemnotWrD0**. Data bit zero of the least significant byte of each ROM word then provides the configuration data stream. By switching **MemConfig** between various data bus lines up to 32 configurations can be stored in ROM, one per bit of the data bus. Connecting **MemConfig** to **GND** gives all **Tstates** configured to four periods; **notMemS1** pulse of maximum duration; **notMemS2-4** delayed by maximum; refresh interval 72 periods of **ClockIn**; refresh enabled; late write.

The external memory configuration table 5.11 shows the contribution of each memory address to the 13 configuration fields. The lowest 12 words (#7FFFFF6C to #7FFFFF98, fields 1 to 6) define the number of extra periods **Tm** to be added to each **Tstate**. If field 2 is 3 then three extra periods will be added to **T2** to extend it to the maximum of four periods.

The next five addresses (field 7) define the duration of **notMemS1** and the following fifteen (fields 8 to 10) define the delays before strobes **notMemS2-4** become active. The five bits allocated to each strobe allow durations of from 0 to 31 periods **Tm**.

Addresses #7FFFFFEC to #7FFFFFF4 (fields 11 and 12) define the refresh interval and whether refresh is to be used, whilst the final address (field 13) supplies a high bit to **MemConfig** if a late write cycle is required.

The columns to the right of the coding table show the values of each configuration bit for the four sample external configuration diagrams. Note the inclusion of period **E** at the end of **T6** in some diagrams. This is inserted to bring the start of the next **Tstate T1** to coincide with a rising edge of **ProcClockOut** (page 178).

Wait states **W** have been added to show the effect of them on strobe timing; they are not part of a configuration. In each case which includes wait states, two wait periods are defined. This shows that if a wait state would cause the start of **T5** to coincide with a falling edge of **ProcClockOut**, another period **Tm** is generated by the EMI to force it to coincide with a rising edge of **ProcClockOut**. This coincidence is only necessary if wait states are added, otherwise coincidence with a falling edge is permitted. Any configuration memory access is only permitted to be extended using wait, up to a total of 14 **ClockIn** periods.

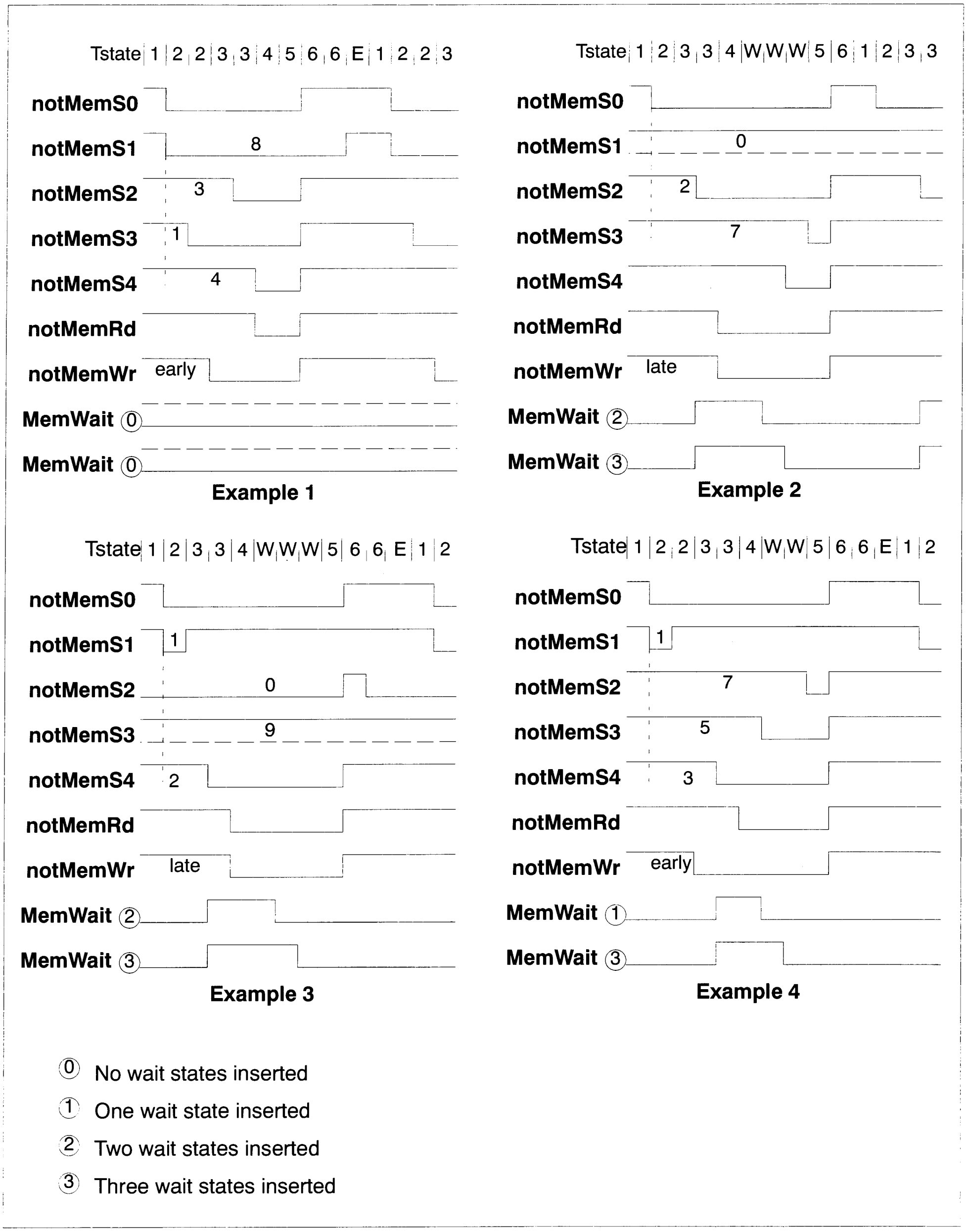
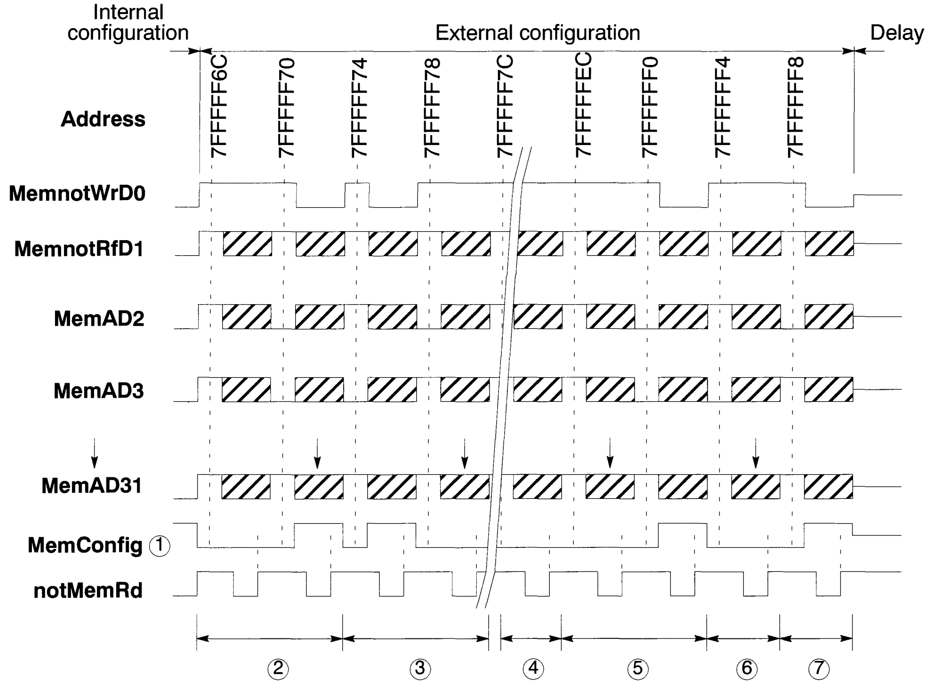


Figure 5.23 IMS T426 external configuration



① **MemConfig** connected to inverse of **MemnotWrD0**

② Configuration field 1; **T1** configured for 2 periods **Tm**

③ Configuration field 2; **T2** configured for 2 periods **Tm**

④ Configuration field 10; most significant bit of **notMemS4** configured high

⑤ Configuration field 11; refresh interval configured for 36 periods **ClockIn**

⑥ Configuration field 12; refresh enabled

⑦ Configuration field 13; early write cycle

Figure 5.24 IMS T426 external configuration scan

Scan cycle	MemAD address	Field	Function	Example diagram			
				1	2	3	4
1	7FFFFF6C	1	T1 least significant bit	0	0	0	0
2	7FFFFF70	1	T1 most significant bit	0	0	0	0
3	7FFFFF74	2	T2 least significant bit	1	0	0	1
4	7FFFFF78	2	T2 most significant bit	0	0	0	0
5	7FFFFF7C	3	T3 least significant bit	1	1	1	1
6	7FFFFF80	3	T3 most significant bit	0	0	0	0
7	7FFFFF84	4	T4 least significant bit	0	0	0	0
8	7FFFFF88	4	T4 most significant bit	0	0	0	0
9	7FFFFF8C	5	T5 least significant bit	0	0	0	0
10	7FFFFF90	5	T5 most significant bit	0	0	0	0
11	7FFFFF94	6	T6 least significant bit	1	0	1	1
12	7FFFFF98	6	T6 most significant bit	0	0	0	0
13	7FFFFF9C	7	notMemS1 least significant bit	0	0	1	1
14	7FFFFFA0	7		0	0	0	0
15	7FFFFFA4	7	⇓ ⇓	0	0	0	0
16	7FFFFFA8	7		1	0	0	0
17	7FFFFFAC	7	notMemS1 most significant bit	0	0	0	0
18	7FFFFFB0	8	notMemS2 least significant bit	1	0	0	1
19	7FFFFFB4	8		1	1	0	1
20	7FFFFFB8	8	⇓ ⇓	0	0	0	1
21	7FFFFFBC	8		0	0	0	0
22	7FFFFFC0	8	notMemS2 most significant bit	0	0	0	0
23	7FFFFFC4	9	notMemS3 least significant bit	1	1	1	1
24	7FFFFFC8	9		0	1	0	0
25	7FFFFFCC	9	⇓ ⇓	0	1	0	1
26	7FFFFFD0	9		0	0	1	0
27	7FFFFFD4	9	notMemS3 most significant bit	0	0	0	0
28	7FFFFFD8	10	notMemS4 least significant bit	0	0	0	1
29	7FFFFFDC	10		0	1	1	1
30	7FFFFFE0	10	⇓ ⇓	1	1	0	0
31	7FFFFFE4	10		0	0	0	0
32	7FFFFFE8	10	notMemS4 most significant bit	0	0	0	0
33	7FFFFFEC	11	Refresh Interval least significant bit	-	-	-	-
34	7FFFFFF0	11	Refresh Interval most significant bit	-	-	-	-
35	7FFFFFF4	12	Refresh Enable	-	-	-	-
36	7FFFFFF8	13	Late Write	0	1	1	0

Table 5.11 IMS T426 external configuration coding

Refresh interval	Interval in μs	Field 11 encoding	Complete cycle (ms)
18	3.6	00	0.922
36	7.2	01	1.843
54	10.8	10	2.765
72	14.4	11	3.686

Table 5.12 IMS T426 memory refresh configuration coding

Refresh intervals are in periods of **ClockIn** and **ClockIn** frequency is 5 MHz:

Interval = 18 * 200 = 3600 ns

Refresh interval is between successive incremental refresh addresses.
Complete cycles are shown for 256 row DRAMS.

Symbol	Parameter	T426–20		T426–25		Units	Notes
		Min	Max	Min	Max		
TMCVRdH	**MemConfig** data setup	25		20		ns	
TRdHMCX	**MemConfig** data hold	0		0		ns	
TS0LRdH	**notMemS0** to configuration data read	388	412	308	332	ns	

Table 5.13 Memory configuration timing

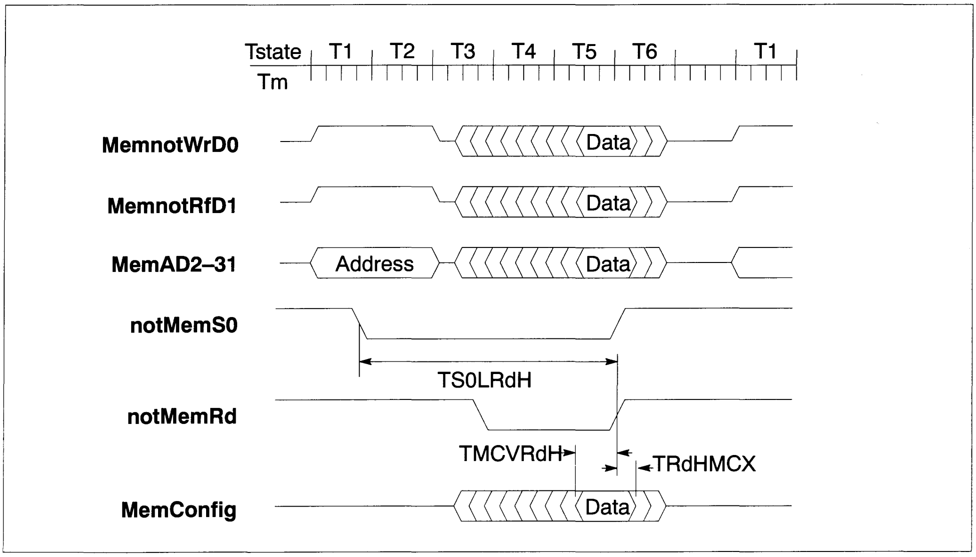


Figure 5.25 IMS T426 external configuration read cycle timing

6 Events

EventReq and **EventAck** provide an asynchronous handshake interface between an external event and an internal process. When an external event takes **EventReq** high the external event channel (additional to the external link channels) is made ready to communicate with a process. When both the event channel and the process are ready the processor takes **EventAck** high and the process, if waiting, is scheduled. **EventAck** is removed after **EventReq** goes low.

EventWaiting is asserted high by the transputer when a process executes an input on the event channel; typically with the occam `EVENT ? ANY` instruction. It remains high whilst the transputer is waiting for or servicing **EventReq** and is returned low when **EventAck** goes high. The **EventWaiting** pin changes near the falling edge of **ProcClockOut** and can therefore be sampled by the rising edge of **ProcClockOut**.

The **EventWaiting** pin can only be asserted by executing an *in* instruction on the event channel. The **EventWaiting** pin is not asserted high when an enable channel (*enbc*) instruction is executed on the Event channel (during an ALT construct in occam, for example). The **EventWaiting** pin can be asserted by executing the occam input on the event channel (such as `EVENT ? ANY`), provided that this does not occur as a guard in an alternative process. The **EventWaiting** pin can not be used to signify that an alternative process (ALT) is waiting on an input from the event channel.

EventWaiting allows a process to control external logic; for example, to clock a number of inputs into a memory mapped data latch so that the event request type can be determined.

Only one process may use the event channel at any given time. If no process requires an event to occur **EventAck** will never be taken high. Although **EventReq** triggers the channel on a transition from low to high, it must not be removed before **EventAck** is high. **EventReq** should be low during **Reset**; if not it will be ignored until it has gone low and returned high. **EventAck** is taken low when **Reset** occurs.

If the process is a high priority one and no other high priority process is running, the latency is as described on page 32. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

Cycle 1 Sample **EventReq** at pad on the rising edge of **ProcClockOut** and synchronize.

Cycle 2 Edge detect the synchronized **EventReq** and form the interrupt request.

Cycle 3 Sample interrupt vector for microcode ROM in the CPU.

Cycle 4 Execute the interrupt routine for the event rather than the next instruction.

		T426-20		T426-25			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TVHKH	**EventReq** response	0		0		ns	1
TKHVL	**EventReq** hold	0		0		ns	1
TVLKL	Delay before removal of **EventAck**	0	157	0	127		
TKLVH	Delay before re-assertion of **EventReq**	0		0		ns	1
TKHEWL	**EventAck** to end of **EventWaiting**	0		0		ns	1

Notes

1 Guaranteed, but not tested.

Table 6.1 Event timing

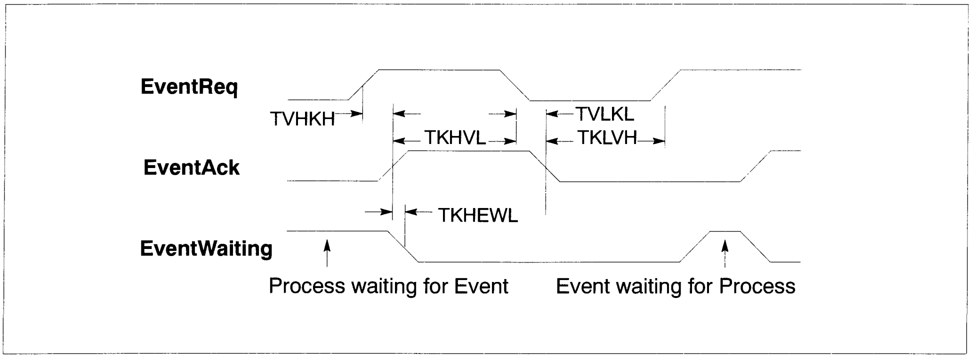


Figure 6.1 IMS T426 event timing

7 Links

Four identical INMOS bi-directional serial links provide synchronized communication between processors and with the outside world. Each link comprises an input channel and output channel. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

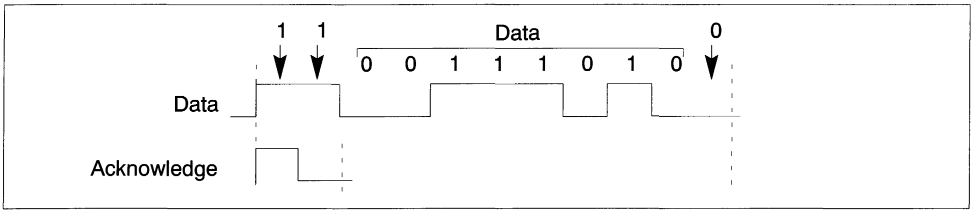


Figure 7.1 IMS T426 link data and acknowledge packets

The IMS T426 links allow an acknowledge packet to be sent before the data packet has been fully received. This overlapped acknowledge technique is fully compatible with all other INMOS transputer links.

The IMS T426 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 5 or 20 Mbits/sec. Links are not synchronized with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimeters. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**, see figure 7.5. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions, see figure 7.6 . If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

Link speeds can be set by **LinkSpecial, Link0Special** and **Link123Special**. The link 0 speed can be set independently. Table 7.1 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed and as **Link123Special** for the others. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

Link Special	Linkn Special	Mbits/sec	Kbytes/sec	
			Uni	Bi
0	0	10	910	1250
0	1	5	450	670
1	0	10	910	1250
1	1	20	1740	2350

Table 7.1 Speed settings for transputer links

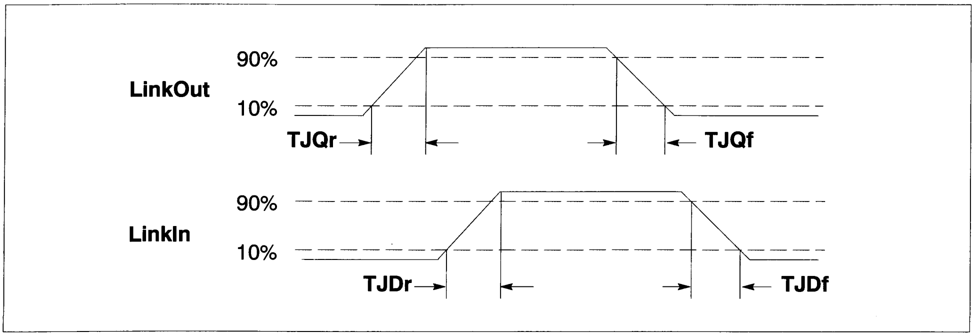


Figure 7.2 IMS T426 link timing

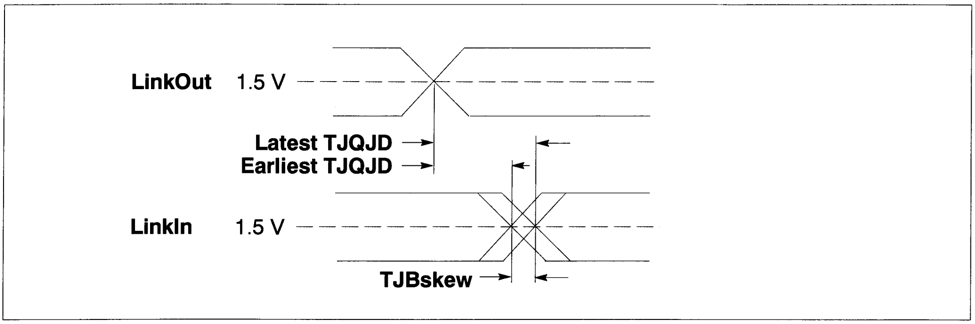


Figure 7.3 IMS T426 buffered link timing

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	1
TJDf	**LinkIn** fall time				20	ns	1
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
		5 Mbits/s			30	ns	2
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		ohms	

Notes

1 Guaranteed, but not tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 7.2 Link timing

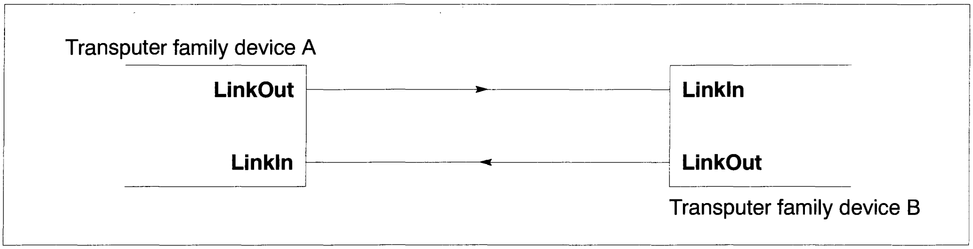


Figure 7.4 IMS T426 links directly connected

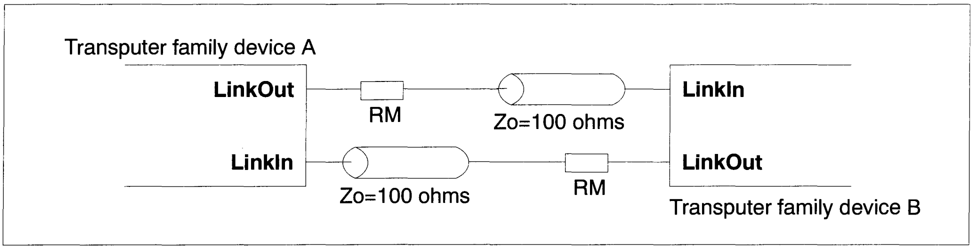


Figure 7.5 IMS T426 links connected by transmission line

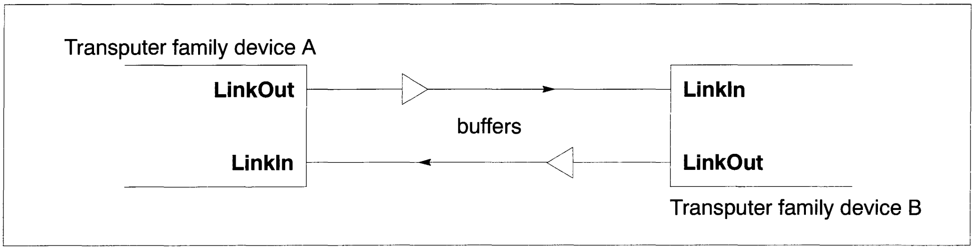


Figure 7.6 IMS T426 links connected by buffers

8 Electrical specifications

8.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1, 2, 3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1, 2, 3
II	Input current		±25	mA	4
tOSC	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 8.1 Absolute maximum ratings

8.2 Operating conditions

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1, 2
CL	Load capacitance on any pin		60	pF	
TA	Operating temperature range	0	70	°C	3

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Air flow rate 400 linear ft/min transverse air flow.

Table 8.2 Operating conditions

8.3 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VIH	High level input voltage	2.0	VDD+0.5	V	1, 2
VIL	Low level input voltage	–0.5	0.8	V	1, 2
II	Input current @ GND<VI<VDD		±10	μA	1, 2
VOH	Output high voltage @ IOH=2mA	VDD–1		V	1, 2
VOL	Output low voltage @ IOL=4mA		0.4	V	1, 2
IOZ	Tristate output current @ GND<V0<VDD		±10	μA	1, 2
PD	Power dissipation		1	W	2, 3
CIN	Input capacitance @ f=1MHz		7	pF	4
COZ	Output capacitance @ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

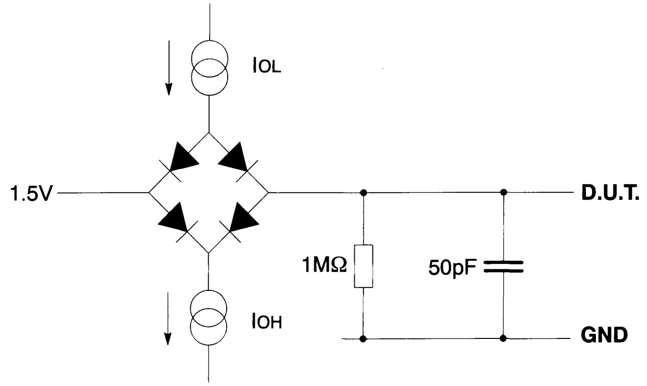
2 Parameters for IMS T426-S measured at 4.75V<**VDD**<5.25V and 0ºC<**TA**<70ºC.
Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution.
Power dissipation for processor operating at 20 MHz.

4 This parameter is sampled and not 100% tested.

Table 8.3 DC characteristics

8.4 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 8.1 Load circuit for AC measurements

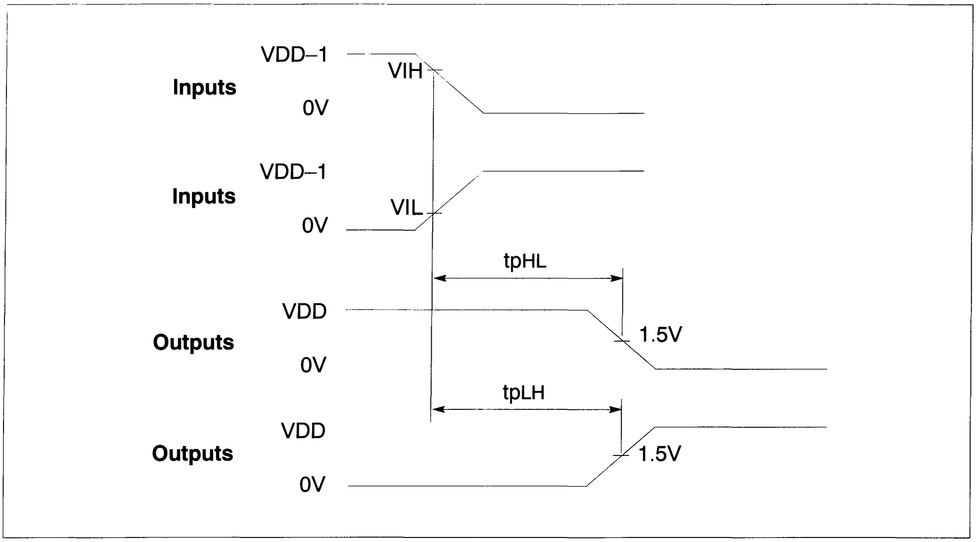


Figure 8.2 AC measurements timing waveforms

8.5 AC timing characteristics

Symbol	Parameter	Min	Max	Units	Notes
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3
TQr	Output rising edges		25	ns	1,3
TQf	Output falling edges		15	ns	1,3
TS0LaX	Address hold after **notMemS0**	**a**–8	**a**+8	ns	4

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 Guaranteed, but not tested.

4 **a** is **T2** where **T2** can be from one to four periods **Tm** in length.
Address lines include **MemnotWrD0**, **MemnotRfD1**, **MemAD2-31**.

Table 8.4 Input edges

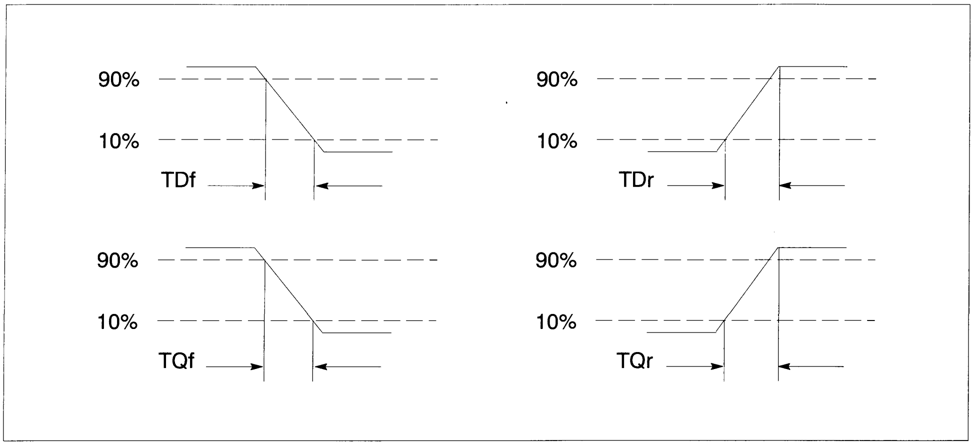


Figure 8.3 IMS T426 input and output edge timing

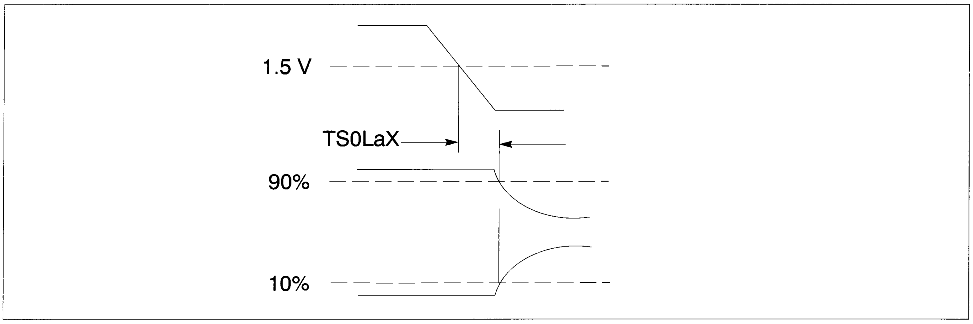
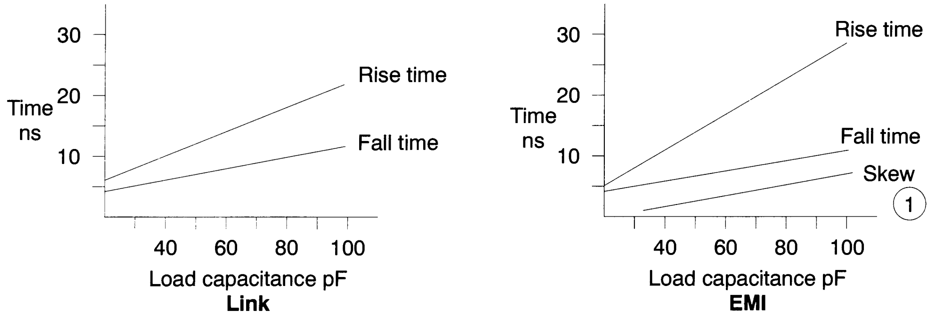


Figure 8.4 IMS T426 tristate timing relative to **notMemS0**



Notes

1 Skew is measured between **ProcClockOut** with a load of 2 Schottky TTL inputs and 30pF and **notMemS0** with a load of 2 Schottky TTL inputs and varying capacitance.

Figure 8.5 Typical rise/fall times

8.6 Power rating

Internal power dissipation (P_{INT}) of transputer and peripheral chips depends on **VDD**, as shown in figure 8.6. P_{INT} is substantially independent of temperature.

Total power dissipation (P_D) of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C (see table 8.1) and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

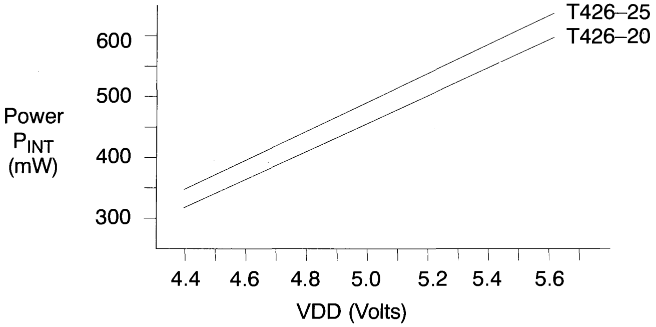
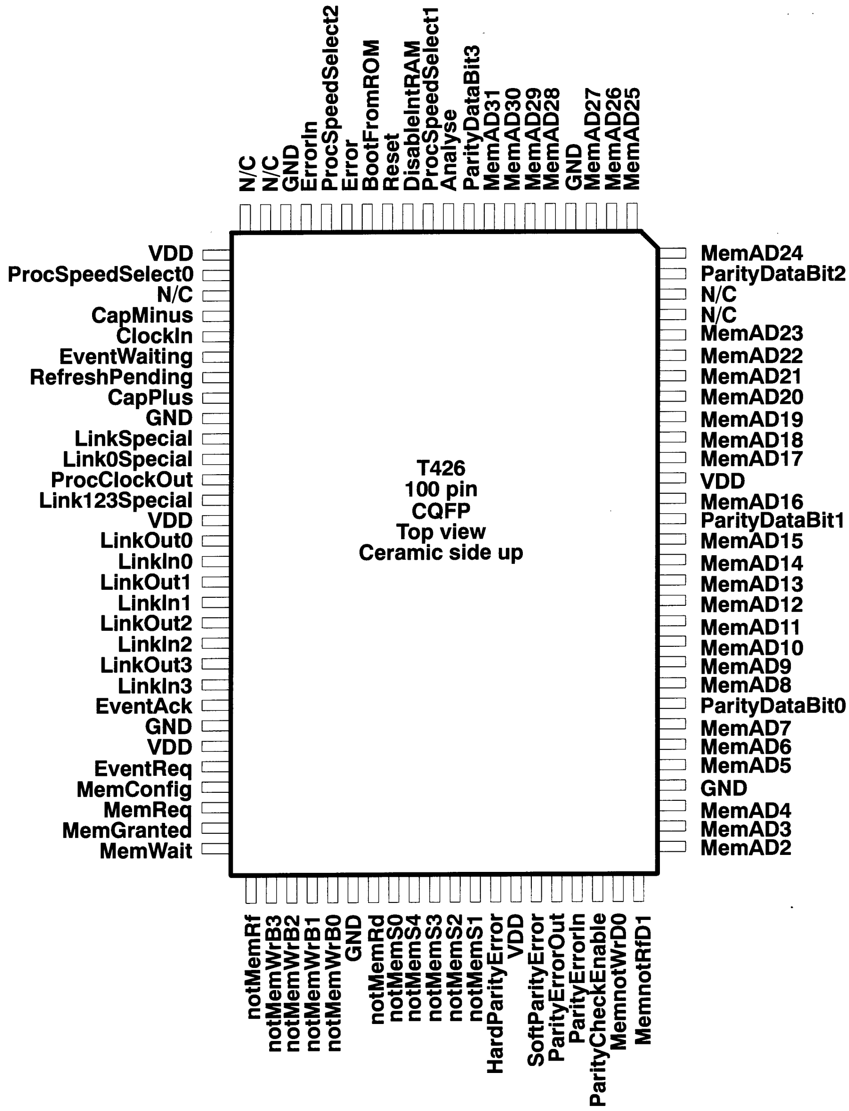


Figure 8.6 IMS T426 internal power dissipation vs VDD

9 Package pinouts

9.1 100 pin cavity-down ceramic quad flat pack package



N/C indicates pins not connected

Figure 9.1 IMS T426 100 pin cavity–down ceramic quad flat pack package pinout

10 Ordering

This section indicates the designation of speed and package selections for the various devices. Speed of **ClockIn** is 5 MHz for all parts. Transputer processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in the external memory section.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	**Processor clock speed**	**Processor cycle time**	**PLLx**	**Package**
IMS T426-F20S	20.0	50	4.0	100 pin ceramic quad flat pack
IMS T426-F25S	25.0	40	5.0	100 pin ceramic quad flat pack

Table 10.1 IMS T426 ordering details

®

IMS T425 transputer

inmos®

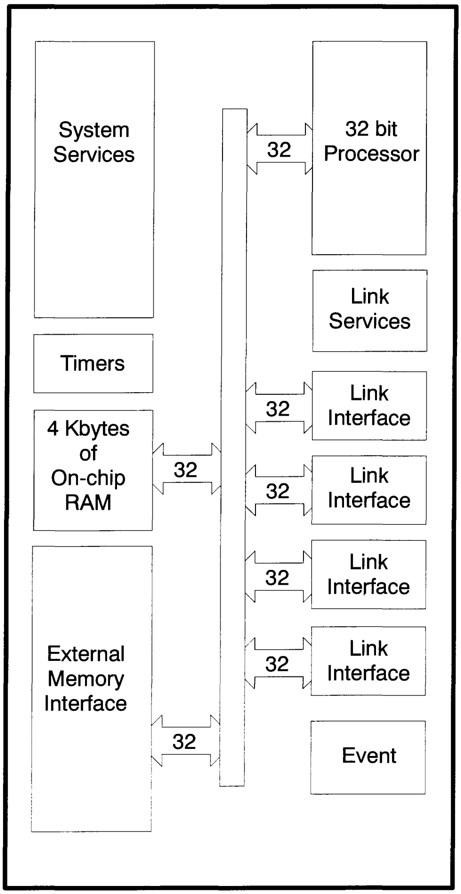
Engineering Data

FEATURES

32 bit architecture
33 ns internal cycle time
30 MIPS peak instruction rate
Pin compatible with IMS T805, IMS T800, IMS T400 and IMS T414
Debugging support
4 Kbytes on-chip static RAM
120 Mbytes/sec sustained data rate to internal memory
4 Gbytes directly addressable external memory
40 Mbytes/sec sustained data rate to external memory
630 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
High performance graphics support with block move instructions
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ± 5% power supply
Packaging 84 pin PGA / 84 pin PLCC / 100 pin CQFP

APPLICATIONS

High speed multi processor systems
High performance graphics processing
Supercomputers
Workstations and workstation clusters
Digital signal processing
Accelerator processors
Distributed databases
System simulation
Telecommunications
Robotics
Fault tolerant systems
Image processing
Pattern recognition
Artificial intelligence



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992
42 1426 03

1 Introduction

The IMS T425 transputer is a 32 bit CMOS microcomputer with graphics support. It has 4 Kbytes on-chip RAM for high speed processing, a configurable memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages and provides direct support for the occam model of concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond.

For convenience of description, the IMS T425 operation is split into the basic blocks shown in figure 1.1.

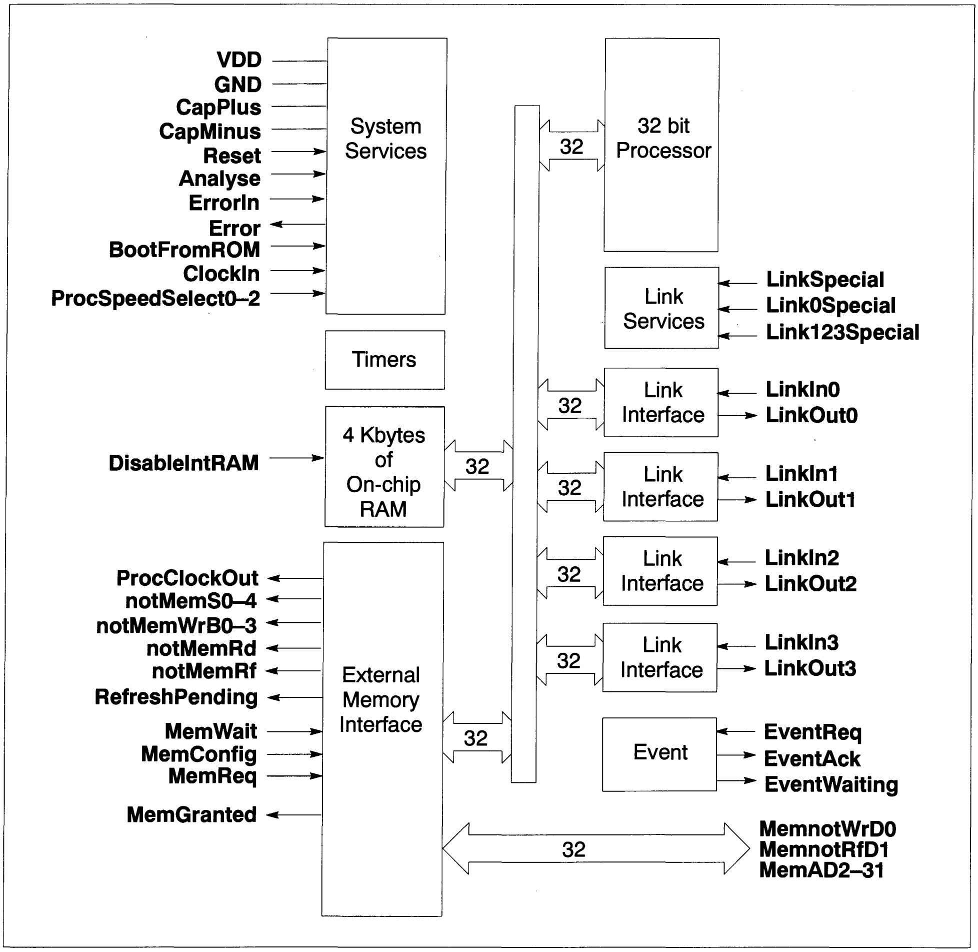


Figure 1.1 IMS T425 block diagram

The processor speed of a device can be pin-selected in stages from 20 MHz up to the maximum allowed for the part. A device running at 30 MHz achieves an instruction throughput of 30 MIPS peak and 15 MIPS sustained.

High performance graphics support is provided by microcoded block move instructions which operate at the speed of memory. The two-dimensional block move instructions provide for contiguous block moves as well as block copying of either non-zero bytes of data only or zero bytes only. Block move instructions can be used to provide graphics operations such as text manipulation, windowing, panning, scrolling and screen updating.

Cyclic redundancy checking (CRC) instructions are available for use on arbitrary length serial data streams, to provide error detection where data integrity is critical. Another feature of the IMS T425, useful for pattern recognition, is the facility to count bits set in a word.

The IMS T425 can directly access a linear address space of 4 Gbytes. The 32 bit wide memory interface uses multiplexed data and address lines and provides a data rate of up to 4 bytes every 100 nanoseconds (40 Mbytes/sec) for a 30 MHz device. A configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems.

System Services include processor reset and bootstrap control, together with facilities for error analysis. Error signals may be daisy-chained in multi-transputer systems.

The standard INMOS communication links allow networks of transputer family products to be constructed by direct point to point connections with no external logic. The IMS T425 links support the standard operating speed of 10 Mbits/sec, but also operate at 5 or 20 Mbits/sec. Each link can transfer data bi-directionally at up to 2.35 Mbytes/sec.

The IMS T425 is pin compatible with the IMS T800 and can be plugged directly into a circuit designed for that device. It has a number of additions to improve hardware interfacing and to facilitate software initialising and debugging. The improvements have been made in an upwards-compatible manner. Software should be recompiled, although no changes to the source code are necessary.

The IMS T425-20 is also pin compatible with the IMS T414-20, as the extra inputs used are all held to ground on the IMS T414. The IMS T425-20 can thus be plugged directly into a circuit designed for a 20 MHz version of the IMS T414.

The transputer is designed to implement the occam language, detailed in the *occam Reference Manual*, but also efficiently supports other languages such as C, Pascal and Fortran. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*. The instruction set of the IMS T425 is a superset of the IMS T800, excluding the FPU instructions. Additional instructions are listed in Chapter 4.

This datasheet supplies hardware implementation and characterization details for the IMS T425. It is intended to be read in conjunction with the Transputer Architecture chapter, which details the architecture of the transputer and gives an overview of occam.

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high.
Pinout details are given on page 267.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
ProcSpeedSelect0–2	in	Processor speed selectors
Reset	in	System reset
Error	out	Error indicator
ErrorIn	in	Error daisychain input
Analyse	in	Error analysis
BootFromROM	in	Boot from external ROM or from link
DisableIntRAM	in	Disable internal RAM

Table 2.1 IMS T425 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemnotWrD0	in/out	Multiplexed data bit 0 and write cycle warning
MemnotRfD1	in/out	Multiplexed data bit 1 and refresh warning
MemAD2–31	in/out	Multiplexed data and address bus
notMemRd	out	Read strobe
notMemWrB0–3	out	Four byte-addressing write strobes
notMemS0–4	out	Five general purpose strobes
notMemRf	out	Dynamic memory refresh indicator
RefreshPending	out	Dynamic memory refresh cycle is pending
MemWait	in	Memory cycle extender
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted
MemConfig	in	Memory configuration data input

Table 2.2 IMS T425 programmable memory interface

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge
EventWaiting	out	Event input requested by software

Table 2.3 IMS T425 event

Pin	In/Out	Function
LinkIn0–3	in	Four serial data input channels
LinkOut0–3	out	Four serial data output channels
LinkSpecial	in	Select non–standard speed as 5 or 20 Mbits/sec
Link0Special	in	Select special speed for Link 0
Link123Special	in	Select special speed for Links 1, 2, 3

Table 2.4 IMS T425 link

3 System services

System services include all the necessary logic to initialize and sustain operation of the device. They also include error handling and analysis facilities.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimize inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1 μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 10 MHz. If a polarized capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

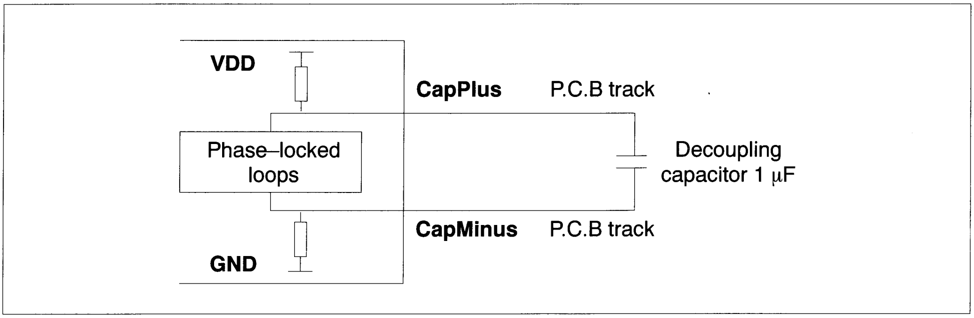


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	T425-20, -25, -30			Units	Notes
		Min	Nom	Max		
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1,2,4
TDCerror	**ClockIn** timing error		.	±0.5	ns	1,3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1,4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 8.3).

Table 3.1 Input clock timing

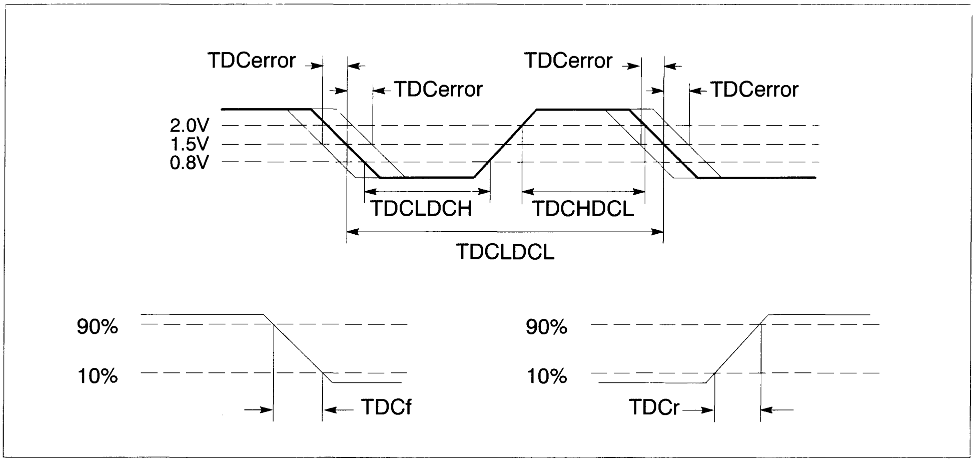


Figure 3.2 **ClockIn** timing

3.4 ProcSpeedSelect0–2

Processor speed of the IMS T425 is variable in discrete steps. The desired speed can be selected, up to the maximum rated for a particular component, by the three speed select lines **ProcSpeedSelect0-2**. The pins are tied high or low, according to table 3.2, for the various speeds. The pins are arranged so that the IMS T425 can be plugged directly into a board designed for an IMS T805.

Only six of the possible speed select combinations are currently used; the other two are not valid speed selectors. The frequency of **ClockIn** for the speeds given in the table is 5 MHz.

ProcSpeed-Select2	ProcSpeed-Select1	ProcSpeed-Select0	Processor Clock Speed MHz	Processor Cycle Time ns	Notes
0	0	0	20.0	50.0	
0	0	1	22.5	44.4	Not supported
0	1	0	25.0	40.0	
0	1	1	30.0	33.3	
1	0	0	35.0	28.6	Not supported
1	0	1			Invalid
1	1	0	17.5	57.1	Not supported
1	1	1			Invalid

Note: Inclusion of a speed selection in this table does not imply immediate availability.

Table 3.2 Processor speed selection

3.5 Bootstrap

The transputer can be bootstrapped either from a link or from external ROM. To facilitate debugging, **BootFromROM** may be dynamically changed but must obey the specified timing restrictions. It is sampled once only by the transputer, before the first instruction is executed after **Reset** is taken low.

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFFFFFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the **W** register points to **MemStart** (page 229).

If **BootFromROM** is connected low (e.g. to **GND**) the transputer will wait for the first bootstrap message to arrive on any one of its links. The transputer is ready to receive the first byte on a link within two processor cycles **TPCLPCL** after **Reset** goes low.

If the first byte received (the control byte) is greater than 1 it is taken as the quantity of bytes to be input. The following bytes, to that quantity, are then placed in internal memory starting at location **MemStart**. Following reception of the last byte the transputer will start executing code at **MemStart** as a low priority process. **BootFromROM** may be taken high after reception of the last byte, if required. The memory space immediately above the loaded code is used as work space. A byte arriving on other links after the control byte has been received and on the bootstrapping link after the last bootstrap byte, will be retained and no acknowledge will be sent until a process inputs from them.

3.6 Peek and poke

Any location in internal or external memory can be interrogated and altered when the transputer is waiting for a bootstrap from link. If the control byte is 0 then eight more bytes are expected on the same link. The first four byte word is taken as an internal or external memory address at which to *poke* (write) the second four byte word. If the control byte is 1 the next four bytes are used as the address from which to *peek* (read) a word of data; the word is sent down the output channel of the same link.

Following such a *peek* or *poke*, the transputer returns to its previously held state. Any number of accesses may be made in this way until the control byte is greater than 1, when the transputer will commence reading its bootstrap program. Any link can be used, but addresses and data must be transmitted via the same link as the control byte.

3.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initializes the transputer, triggers the memory configuration sequence and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) is ignored whilst **Reset** is high but can occur before bootstrap (page 246).

After the end of **Reset** there will be a delay of 144 periods of **ClockIn** (figure 3.3). Following this, the **MemnotWrD0**, **MemnotRfD1** and **MemAD2-31** pins will be scanned to check for the existence of a preprogrammed memory interface configuration (page 249). This lasts for a further 144 periods of **ClockIn**. Regardless of whether a configuration was found, 36 configuration read cycles will then be performed on external memory using the default memory configuration (page 251), in an attempt to access the external configuration ROM. A delay will then occur, its period depending on the actual configuration. Finally eight complete and consecutive refresh cycles will initialize any dynamic RAM, using the new memory configuration. If the memory configuration does not enable refresh of dynamic RAM the refresh cycles will be replaced by an equivalent delay with no external memory activity.

If **BootFromROM** is high bootstrapping will then take place immediately, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

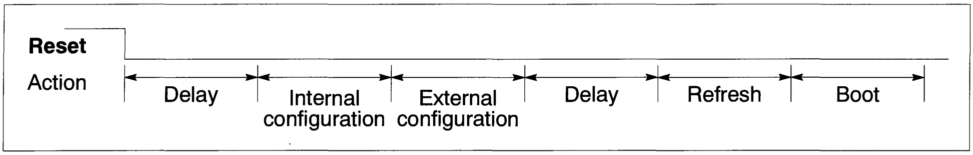


Figure 3.3 IMS T425 post-reset sequence

3.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 48). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags **Error**, **HaltOnError** and **EnableJOBreak** are normally cleared at reset on the IMS T425; however, if **Analyse** is asserted the flags are not altered. Memory refresh continues.

Input links will continue with outstanding transfers. Output links will not make another access to memory for data but will transmit only those bytes already in the link buffer. Providing there is no delay in link acknowledgement, the links should be inactive within a few microseconds of the transputer halting.

Reset should not be asserted before the transputer has halted and link transfers have ceased. When **Reset** is taken low whilst **Analyse** is high, neither the memory configuration sequence nor the block of eight refresh cycles will occur; the previous memory configuration will be used for any external memory accesses. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 3.3.

I	**MemStart** if bootstrapping from a link, or the external memory bootstrap address if bootstrapping from ROM.
W	**MemStart** if bootstrapping from ROM, or the address of the first free word after the bootstrap program if bootstrapping from link.
A	The value of **I** when the processor halted.
B	The value of **W** when the processor halted, together with the priority of the process when the transputer was halted (i.e. the **W** descriptor).
C	The ID of the bootstrapping link if bootstrapping from link.

Table 3.3 Register values after **Analyse**

Symbol	Parameter	T425-20, -25, -30			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

1 Full periods of **ClockIn TDCLDCL** required.

2 At power-on reset.

3 Must be stable until after end of bootstrap period. See Bootstrap section 3.5.

Table 3.4 **Reset** and **Analyse** timing

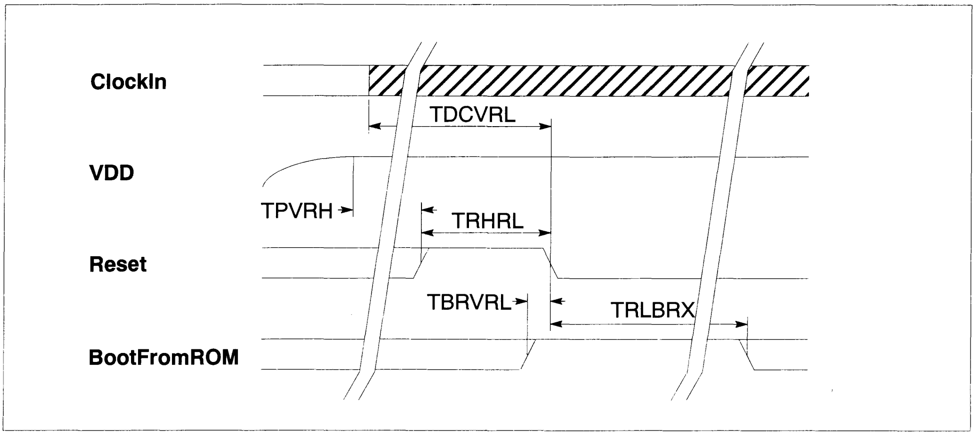


Figure 3.4 Transputer **Reset** timing with **Analyse** low

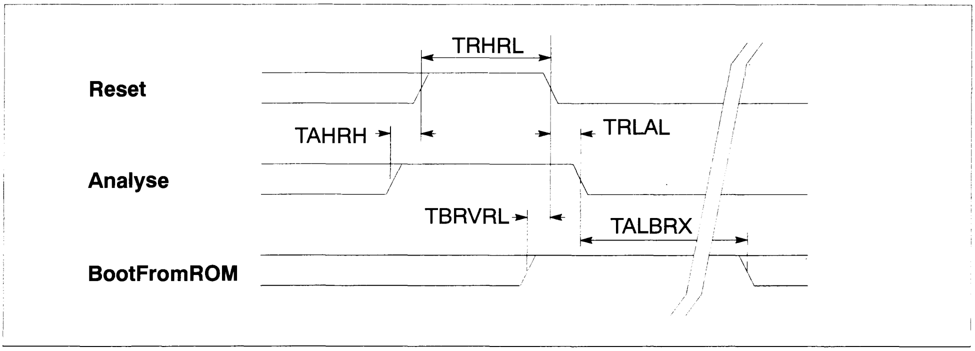


Figure 3.5 **Reset** and **Analyse** timing

3.9 Error, ErrorIn

The **Error** pin carries the OR'ed output of the internal **Error** flag and the **ErrorIn** input. If **Error** is high it indicates either that **ErrorIn** is high or that an error was detected in one of the processes. An internal error can be caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 48). Once set, the **Error** flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (section 3.8).

A process can be programmed to stop if the **Error** flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data. **ErrorIn** does not directly affect the status of a processor in any way.

By setting the **HaltOnError** flag the transputer itself can be programmed to halt if **Error** becomes set. If **Error** becomes set after **HaltOnError** has been set, all processes on that transputer will cease but will not necessarily cause other transputers in a network to halt. Setting **HaltOnError** after **Error** will not cause the transputer to halt; this allows the processor reset and analyse facilities to function with the flags in indeterminate states.

An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by 'daisy-chaining' the **ErrorIn** and **Error** pins of a number of processors and applying the final **Error** output signal to the **EventReq** pin of a suitably programmed master transputer (see figure 3.6). Since the process state is preserved when stopped by an error, the master transputer can then use the analyse function to debug the fault. When using such a circuit, note that the **Error** flag is in an indeterminate state on power up; the circuit and software should be designed with this in mind.

Error checks can be removed completely to optimize the performance of a proven program; any unexpected error then occurring will have an arbitrary undefined effect.

If a high priority process pre-empts a low priority one, status of the **Error** and **HaltOnError** flags is saved for the duration of the high priority process and restored at the conclusion of it. Status of both flags is transmitted to the high priority process. Either flag can be altered in the process without upsetting the error status of any complex operation being carried out by the pre-empted low priority process.

In the event of a transputer halting because of **HaltOnError**, the links will finish outstanding transfers before shutting down. If **Analyse** is asserted then all inputs continue but outputs will not make another access to memory for data. Memory refresh will continue to take place.

After halting due to the **Error** flag changing from 0 to 1 whilst **HaltOnError** is set, register **I** points two bytes past the instruction which set **Error**. After halting due to the **Analyse** pin being taken high, register **I** points one byte past the instruction being executed. In both cases **I** will be copied to register **A**.

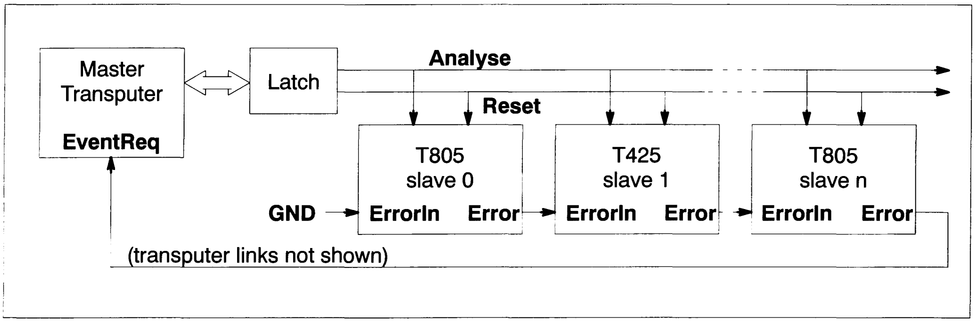


Figure 3.6 Error handling in a multi–transputer system

4 Memory

The IMS T425 has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (page 233). The transputer can also access 4 Gbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

IMS T425 memory is byte addressed, with words aligned on four-byte boundaries. The least significant byte of a word is the lowest addressed byte. The bits in a byte are numbered 0 to 7, with bit 0 the least significant. The bytes are numbered from 0, with byte 0 the least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

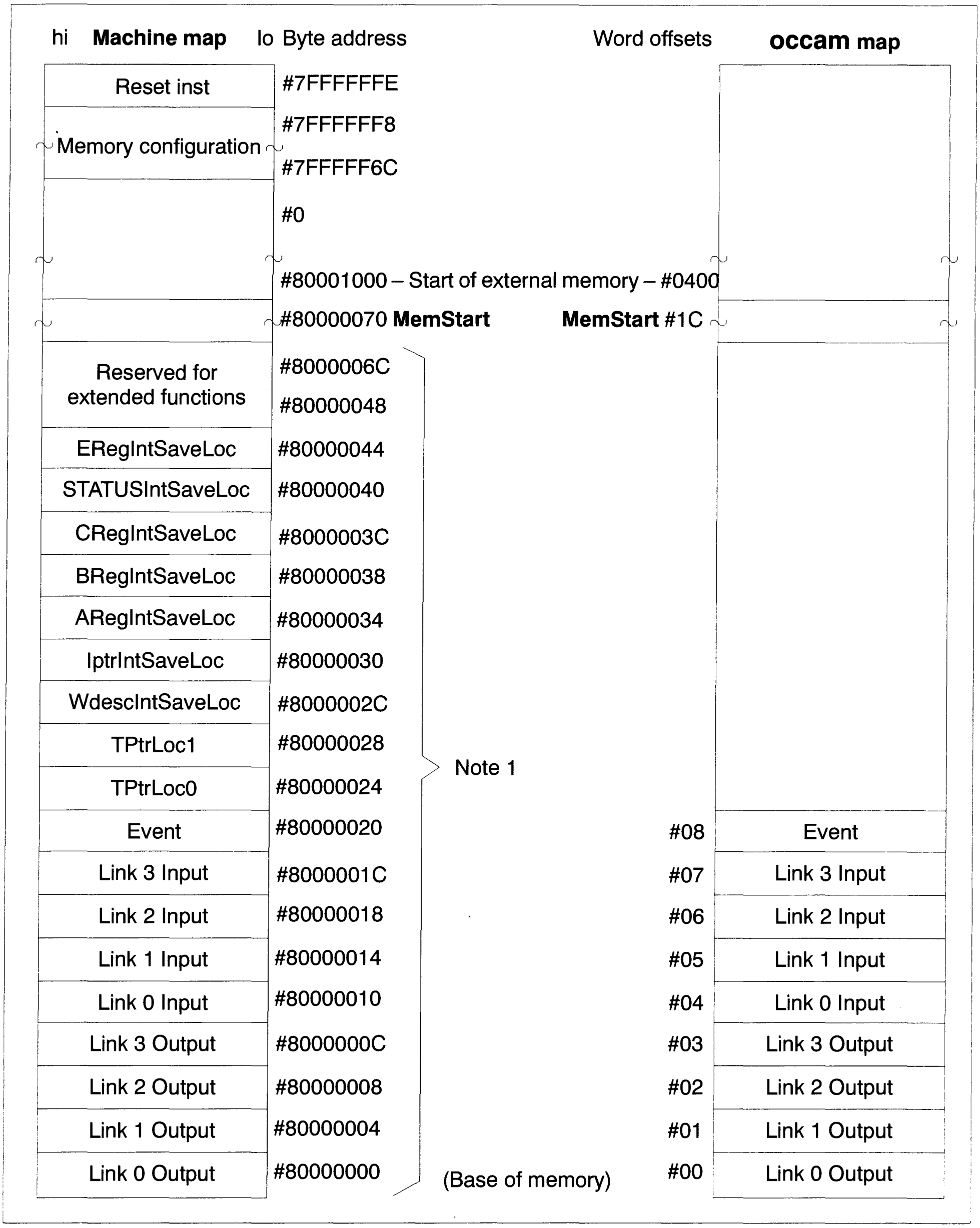
Internal memory starts at the most negative address #80000000 and extends to #80000FFF. User memory begins at #80000070; this location is given the name **MemStart**. An instruction *ldmemstartval* is provided to obtain the value of **MemStart**.

The context of a process in the transputer model involves a workspace descriptor (**WPtr**) and an instruction pointer (**IPtr**). **WPtr** is a word address pointer to a workspace in memory. **IPtr** points to the next instruction to be executed for the process which is the currently executing process. The context switch performed by the breakpoint instruction swaps the **WPtr** and **IPtr** of the currently executing process with the **WPtr** and **IPtr** held above **MemStart**. Two contexts are held above **MemStart**, one for high priority and one for low priority; this allows processes at both levels to have breakpoints. Note that on bootstrapping from a link, these contexts are overwritten by the loaded code. If this is not acceptable, the values should be peeked from memory before bootstrapping from a link. The reserved area of internal memory below **MemStart** is used to implement link and event channels.

Two words of memory are reserved for timer use, **TPtrLoc0** for high priority processes and **TPtrLoc1** for low priority processes. They either indicate the relevant priority timer is not in use or point to the first process on the timer queue at that priority level.

Values of certain processor registers for the current low priority process are saved in the reserved **IntSaveLoc** locations when a high priority process pre-empts a low priority one. Other locations are reserved for extended features such as block moves.

External memory space starts at #80001000 and extends up through #00000000 to #7FFFFFFF. Memory configuration data and ROM bootstrapping code must be in the most positive address space, starting at #7FFFFF6C and #7FFFFFFE respectively. Address space immediately below this is conventionally used for ROM based code.



Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 226). For details see the *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 4.1 IMS T425 memory map

5 External memory interface

The External Memory Interface (EMI) allows access to a 32 bit address space, supporting dynamic and static RAM as well as ROM and EPROM. EMI timing can be configured at **Reset** to cater for most memory types and speeds, and a program is supplied with the Transputer Development System to aid in this configuration.

There are 17 internal configurations which can be selected by a single pin connection (page 104). If none are suitable the user can configure the interface to specific requirements, as shown in page 106.

The external memory cycle is divided into six **Tstates** with the following functions:

T1 Address setup time before address valid strobe.

T2 Address hold time after address valid strobe.

T3 Read cycle tristate or write cycle data setup.

T4 Extendable data setup time.

T5 Read or write data.

T6 Data hold.

Under normal conditions each **Tstate** may be from one to four periods **Tm** long, the duration being set during memory configuration. The default condition on **Reset** is that all **Tstates** are the maximum four periods **Tm** long to allow external initialisation cycles to read slow ROM.

Period **T4** can be extended indefinitely by adding externally generated wait states.

An external memory cycle is always an even number of periods **Tm** in length and the start of **T1** always coincides with a rising edge of **ProcClockOut**. If the total configured quantity of periods **Tm** is an odd number, one extra period **Tm** will be added at the end of **T6** to force the start of the next **T1** to coincide with a rising edge of **ProcClockOut**. This period is designated **E** in configuration diagrams (figure 6.19).

During an internal memory access cycle the external memory interface bus **MemAD2-31** reflects the word address used to access internal RAM, **MemnotWrD0** reflects the read/write operation and **MemnotRfD1** is high; all control strobes are inactive. This is true unless and until a memory refresh cycle or DMA (memory request) activity takes place, when the bus will carry the appropriate external address or data.

The bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 80).

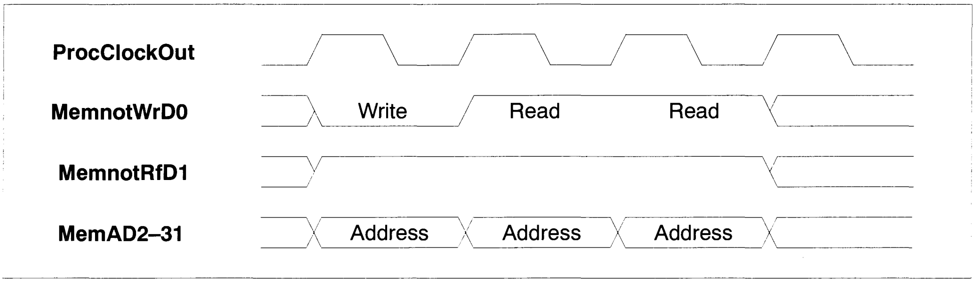


Figure 5.1 IMS 425 bus activity for internal memory cycle

5.1 Pin functions

5.1.1 MemAD2–31

External memory addresses and data are multiplexed on one bus. Only the top 30 bits of address are output on the external memory interface, using pins **MemAD2-31**. They are normally output only during **Tstates T1** and **T2**, and should be latched during this time. The data bus is 32 bits wide. It uses **MemAD2-31** for the top 30 bits and **MemnotRfD1** and **MemnotWrD0** for the lower two bits.

5.1.2 notMemRd

For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Data is read by the transputer on the rising edge of this strobe, and may be removed immediately afterward. If the strobe duration is insufficient it may be extended by adding extra periods **Tm** to either or both of the **Tstates T4** and **T5**. Further extension may be obtained by inserting wait states at the end of **T4**.

5.1.3 MemnotWrD0

During **T1** and **T2** this pin will be low if the cycle is a write cycle, otherwise it will be high. During **Tstates T3** to **T6** it becomes bit 0 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

5.1.4 notMemWrB0–3

Because the transputer uses word addressing, four write strobes are provided; one to write each byte of the word. **notMemWrB0** addresses the least significant byte.

5.1.5 notMemS0–4

To facilitate control of different types of memory and devices, the EMI is provided with five strobe outputs, four of which can be configured by the user. The strobes are conventionally assigned the functions shown in the read and write cycle diagrams, although there is no compulsion to retain these designations.

5.1.6 MemWait

Wait states can be selected by taking **MemWait** high. Externally generated wait states can be added to extend the duration of **T4** indefinitely.

5.1.7 MemnotRfD1

During **T1** and **T2**, this pin is low if the address on **MemAD2-31** is a refresh address, otherwise it is high. During **Tstates T3** to **T6** it becomes bit 1 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

5.1.8 notMemRf

The IMS T425 can be operated with memory refresh enabled or disabled. The selection is made during memory configuration, when the refresh interval is also determined.

5.1.9 RefreshPending

When high, this pin signals that a refresh cycle is pending.

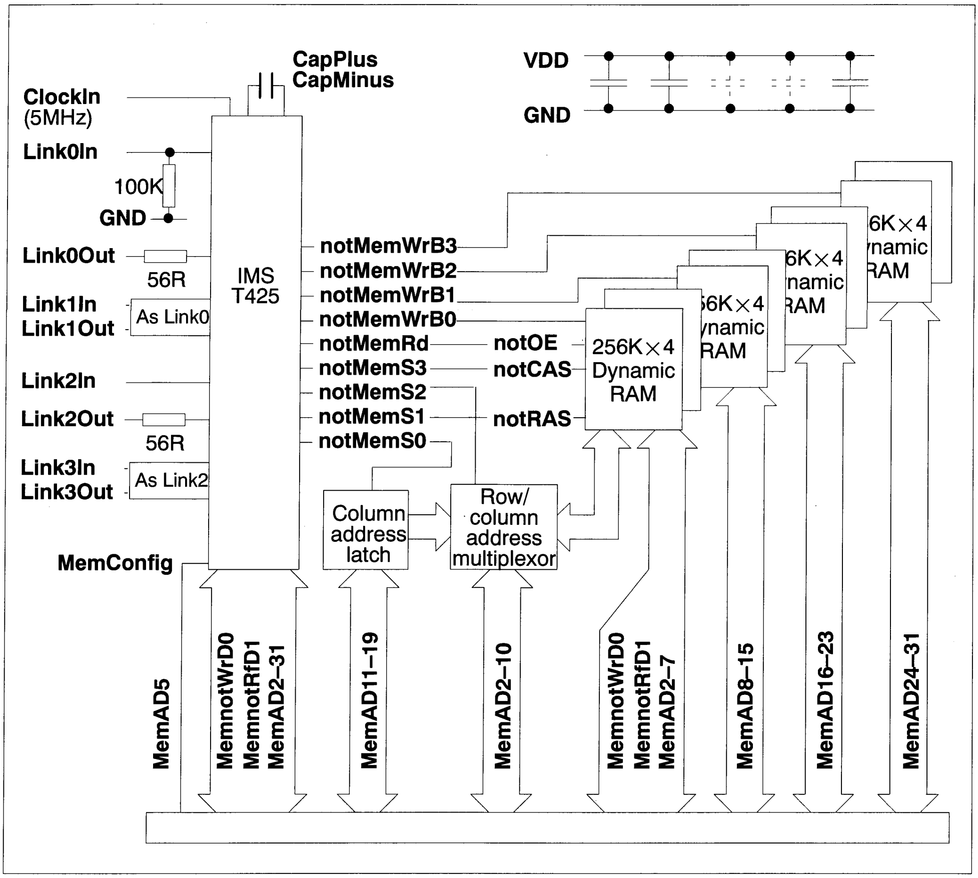


Figure 5.2 IMS T425 application

5.1.10 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high. **MemGranted** follows the timing of the bus being tristated and can be used to signal to the device requesting the DMA that it has control of the bus. Note that **MemGranted** changes on the falling edge of **ProcClockOut** and can therefore be sampled to establish control of the bus on the rising edge of **ProcClockOut**.

5.1.11 MemConfig

MemConfig is an input pin used to read configuration data when setting external memory interface (EMI) characteristics.

5.1.12 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn**. Its period is equal to one internal microcode cycle time, and can be derived from the formula

TPCLPCL = TDCLDCL / PLLx

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (Ordering section).

The time value **Tm** is used to define the duration of **Tstates** and, hence, the length of external memory cycles; its value is exactly half the period of one **ProcClockOut** cycle (0.5***TPCLPCL**), regardless of mark/ space ratio of **ProcClockOut**.

Edges of the various external memory strobes coincide with rising or falling edges of **ProcClockOut**. It should be noted, however, that there is a skew associated with each coincidence. The value of skew depends on whether coincidence occurs when the **ProcClockOut** edge and strobe edge are both rising, when both are falling or if either is rising when the other is falling. Timing values given in the strobe tables show the best and worst cases. If a more accurate timing relationship is required, the exact **Tstate** timing and strobe edge to **ProcClockOut** relationships should be calculated and the correct skew factors applied from the edge skew timing table 6.4.

Symbol	**Parameter**	**T425-20**		**T425-25**		**T425-30**		**Units**	**Notes**
		Min	**Max**	**Min**	**Max**	**Min**	**Max**		
TPCLPCL	**ProcClockOut** period	48	52	38	42	31.5	35	ns	2
TPCHPCL	**ProcClockOut** pulse width high	13.5	28.5	8.5	23.5	13	22	ns	2
TPCLPCH	**ProcClockOut** pulse width low	**a**		**a**		**a**		ns	2, 3, 4
Tm	**ProcClockOut** half cycle	24	26	19	21	15.5	17.5	ns	2
TPCstab	**ProcClockOut** stability		8		8		8	%	1,2

Notes

1 Stability is the variation of cycle periods between two consecutive cycles, measured at corresponding points on the cycles.

2 This parameter is sampled and not 100% tested.

3 **a** is TPCLPCL – TPCHPCL.

4 This is a nominal value.

Table 5.1 ProcClockOut

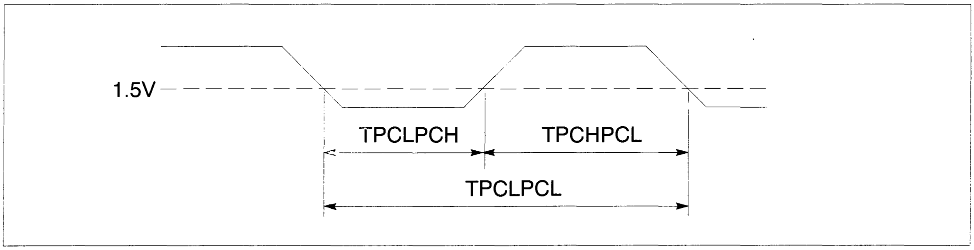


Figure 5.3 IMS T425 ProcClockOut timing

5.2 Read cycle

Byte addressing is carried out internally by the transputer for read cycles. For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Read cycle data may be set up on the data bus at any time after the start of **T3**, but must be valid when the transputer reads it at the end of **T5**. Data may be removed any time during **T6**, but must be off the bus no later than the end of that period.

notMemS0 is a fixed format strobe. Its leading edge is always coincident with the start of **T2** and its trailing edge always coincident with the end of **T5**.

The leading edge of **notMemS1** is always coincident with the start of **T2**, but its duration may be configured to be from zero to 31 periods **Tm**. Regardless of the configured duration, the strobe will terminate no later than the end of **T6**. The strobe is sometimes programmed to extend beyond the normal end of **Tmx**. When wait states are inserted into an EMI cycle the end of **Tmx** is delayed, but the potential active duration of the strobe is not altered. Thus the strobe can be configured to terminate relatively early under certain conditions (page 96). If **notMemS1** is configured to be zero it will never go low.

notMemS2, **notMemS3** and **notMemS4** are identical in operation. They all terminate at the end of **T5**, but the start of each can be delayed from one to 31 periods **Tm** beyond the start of **T2**. If the duration of one of these strobes would take it past the end of **T5** it will stay high. This can be used to cause a strobe to become active only when wait states are inserted. If one of these strobes is configured to zero it will never go low. Figure 6.6 shows the effect of **Wait** on strobes in more detail; each division on the scale is one period **Tm**.

In the read cycle timing diagrams **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**.

		T425-20		T425-25		T425-30			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Note**
TaZdV	Address tristate to data valid	0		0		0		ns	3
TdVRdH	Data setup before read	25		20		15		ns	
TRdHdX	Data hold after read	0		0		0		ns	3
TS0LRdL	**notMemS0** before start of read	**a**–4	**a**+4	**a**–4	**a**+4	**a**–3	**a**+3	ns	1
TS0HRdH	End of read from end of **notMemS0**	–4	4	–4	4	–4	2	ns	
TRdLRdH	Read period	**b**–3	**b**+5	**b**–3	**b**+5	**b**–3	**b**+3	ns	2

Notes

1 **a** is total of **T2**+**T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

2 **b** is total of **T4**+**Twait**+**T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

3 Guaranteed, but not tested.

Table 5.2 Read

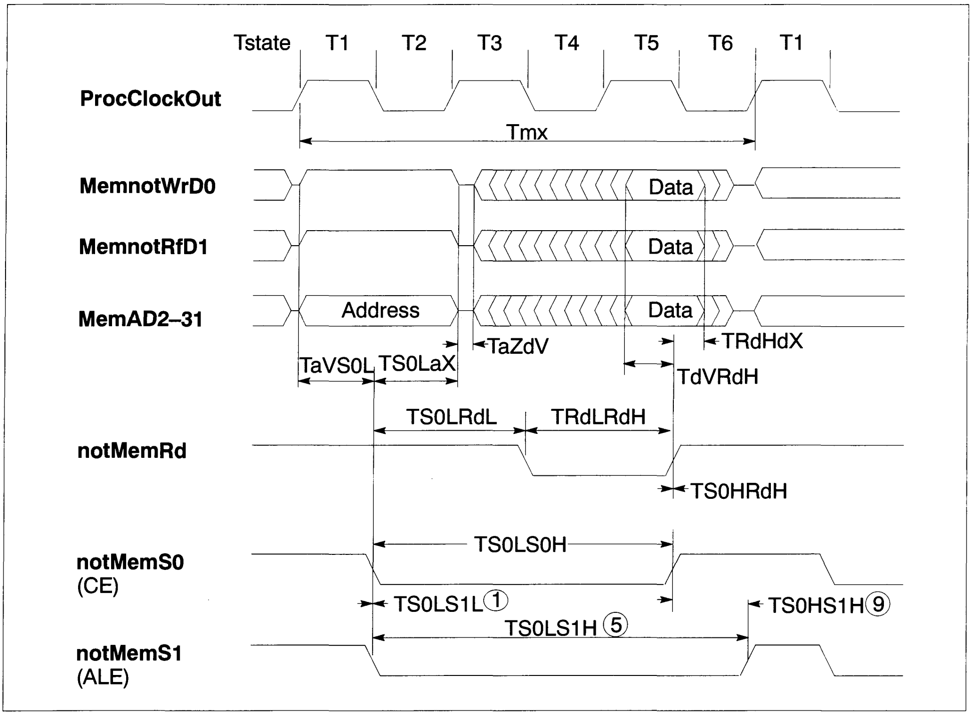


Figure 5.4 IMS T425 external read cycle: static memory

			T425-20		T425-25		T425-30		
Symbol	**n**	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Note**
TaVS0L		Address setup before **notMemS0**	a–8		a–8		a–4		1
TS0LaX		Address hold after **notMemS0**	b–8	b+8	b–8	b+8	b–4	b+5	2
TS0LS0H		**notMemS0** pulse width low	c–5	c+6	c–5	c+6	c–5	c+6	3
TS0LS1L	1	**notMemS1** from **notMemS0**	–4	4	–4	4	–4	4	10
TS0LS1H	5	**notMemS1** end from **notMemS0**	d–1	d+9	d–1	d+9	d–1	d+5	4,6
TS0HS1H	9	**notMemS1** end from **notMemS0** end	e–8	e+4	e–8	e+4	e–5	e+2	5,6
TS0LS2L	2	**notMemS2** delayed after **notMemS0**	f–6	f+5	f–6	f+5	f–4	f+3	7
TS0LS2H	6	**notMemS2** end from **notMemS0**	c–5	c+7	c–5	c+7	c–5	c+3	3,10
TS0HS2H	10	**notMemS2** end from **notMemS0** end	–4	7	–4	7	–4	5	10
TS0LS3L	3	**notMemS3** delayed after **notMemS0**	f–6	f+5	f–6	f+5	f–4	f+3	7
TS0LS3H	7	**notMemS3** end from **notMemS0**	c–5	c+7	c–5	c+7	c–5	c+3	3,10
TS0HS3H	11	**notMemS3** end from **notMemS0** end	–4	7	–4	7	–4	5	10
TS0LS4L	4	**notMemS4** delayed after **notMemS0**	f–6	f+5	f–6	f+5	f–4	f+3	7
TS0LS4H	8	**notMemS4** end from **notMemS0**	c–5	c+7	c–5	c+7	c–5	c+3	3,10
TS0HS4H	12	**notMemS4** end from **notMemS0** end	–4	7	–4	7	–4	5	10
Tmx		Complete external memory cycle							8,9
All timings in nanoseconds (ns).									

Notes

1 **a** is **T1** where **T1** can be from one to four periods **Tm** in length.

2 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

3 **c** is total of **T2**+**T3**+**T4**+**Twait**+**T5** where **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

4 **d** can be from zero to 31 periods **Tm** in length.

5 **e** can be from –27 to +4 periods **Tm** in length.

6 If the configuration would cause the strobe to remain active past the end of **T6** it will go high at the end of **T6**. If the strobe is configured to zero periods **Tm** it will remain high throughout the complete cycle **Tmx**.

7 **f** can be from zero to 31 periods **Tm** in length. If this length would cause the strobe to remain active past the end of **T5** it will go high at the end of **T5**. If the strobe value is zero periods **Tm** it will remain low throughout the complete cycle **T1** to **T5**, going high only for first **Tm** of **T6**.

8 **Tmx** is one complete external memory cycle comprising the total of **T1**+**T2**+**T3**+**T4**+**Twait**+**T5**+**T6** where **T1**, **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length, **T6** can be from one to five periods **Tm** in length and **Twait** may be zero or any number of periods **Tm** in length.

9 Guaranteed, but not tested.

10 Sampled, not 100% tested.

Table 5.3 IMS T425 strobe timing

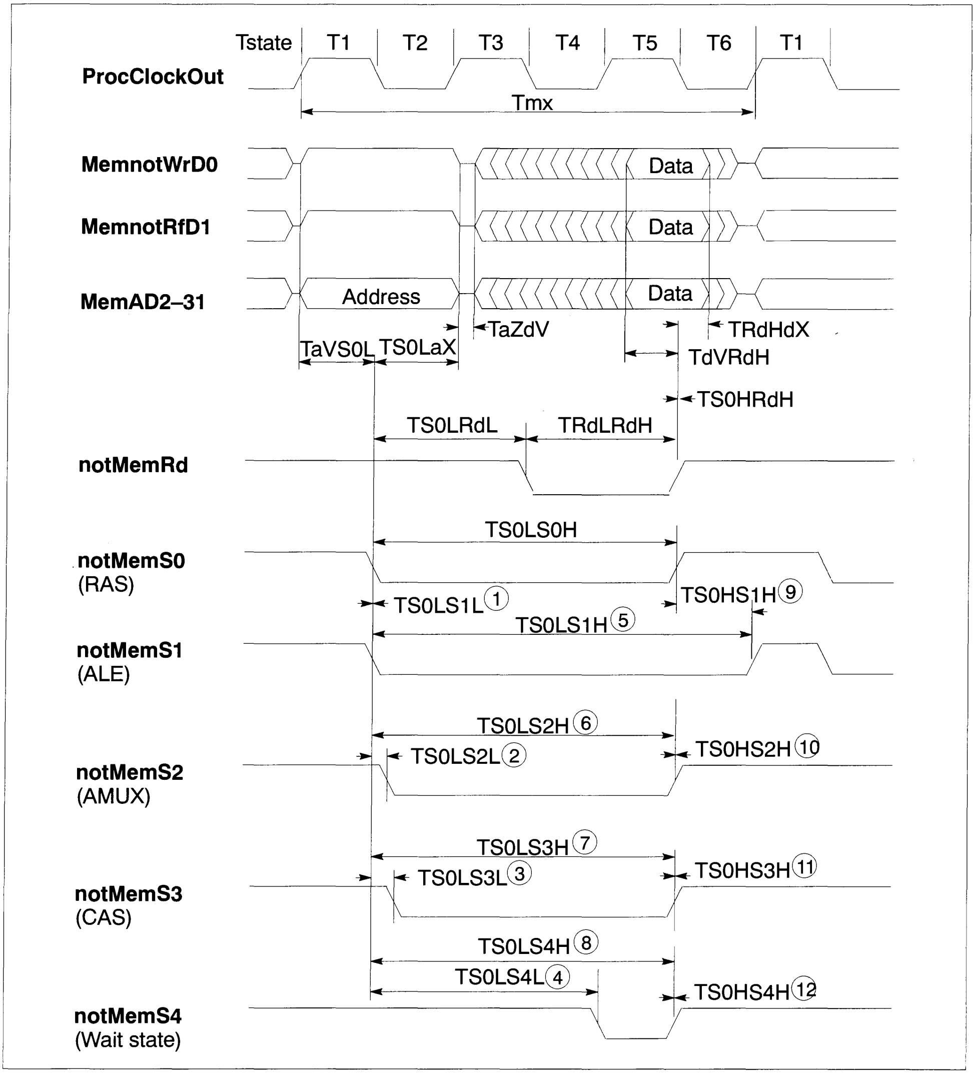


Figure 5.5 IMS T425 external read cycle: dynamic memory

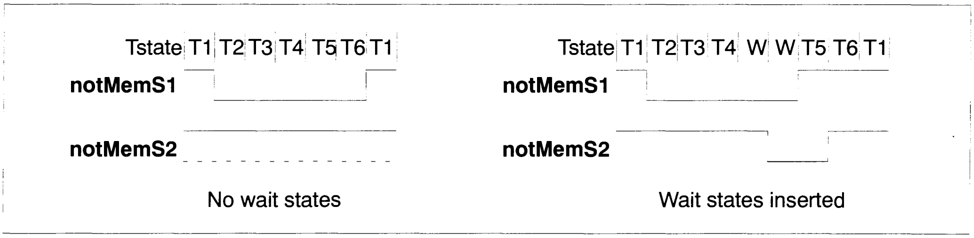


Figure 5.6 IMS T425 effect of wait states on strobes

Symbol	Parameter	T425-20		T425-25		T425-30		Note
		Min	Max	Min	Max	Min	Max	
TPCHS0H	**notMemS0** rising from **ProcClockOut** rising	–6	4	–6	4	–4	3	1
TPCLS0H	**notMemS0** rising from **ProcClockOut** falling	–5	10	–5	10	–3.5	7	1
TPCHS0L	**notMemS0** falling from **ProcClockOut** rising	–8	3	–8	3	–5.5	2	1
TPCLS0L	**notMemS0** falling from **ProcClockOut** falling	–5	7	–5	7	–3.5	5	1
All timings in nanoseconds (ns).								

Notes

1 Sampled, not 100% tested.

Table 5.4 Strobe S0 to **ProcClockOut** skew

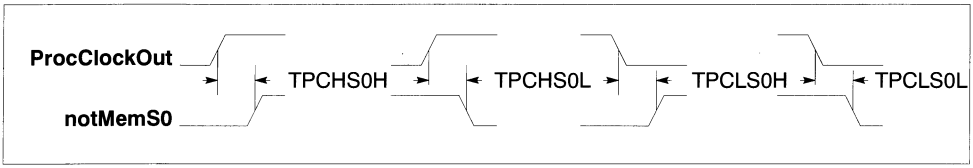


Figure 5.7 IMS T425 skew of **notMemS0** to **ProcClockOut**

5.3 Write cycle

For write cycles the relevant bytes in memory are addressed by the write strobes **notMemWrB0-3**. If a particular byte is not to be written, then the corresponding data outputs are tristated.

For a write cycle pin **MemnotWrD0** will be low during **T1** and **T2**. Write data is placed on the bus at the start of **T3** and removed at the end of **T6**. If **T6** is extended to force the next cycle **Tmx** (page 87) to start on a rising edge of **ProcClockOut**, data will be valid during this time also.

The transputer has both early and late write cycle modes. For a late write cycle the relevant write strobes **notMemWrB0-3** are low during **T4** and **T5**; for an early write they are also low during **T3**. Data should be latched into memory on the rising edge of the strobes in both cases, although it is valid until the end of **T6**. If the strobe duration is insufficient, it may be extended at configuration time by adding extra periods **Tm** to either or both of **Tstates T4** and **T5** for both early and late modes. For an early cycle they may also be added to **T3**. Further extension may be obtained by inserting wait states at the end of **T4**. If the data hold time is insufficient, extra periods **Tm** may be added to **T6** to extend it.

In the write cycle timing diagram **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**. The strobe is inactive during internal memory cycles.

Symbol	Parameter	T425-20		T425-25		T425-30		Notes
		Min	Max	Min	Max	Min	Max	
TdVWrH	Data setup before write	**d**–7	**d**+10	**d**–7	**d**+10	**d**–4	**d**+5	1,5
TWrHdX	Data hold after write	**a**–10	**a**+5	**a**–10	**a**+5	**a**–6	**a**+2	1,2
TS0LWrL	**notMemS0** before start of early write	**b**–5	**b**+5	**b**–5	**b**+5	**b**–5	**b**+2	1,3
	notMemS0 before start of late write	**c**–5	**c**+5	**c**–5	**c**+5	**c**–5	**c**+2	1,4
TS0HWrH	End of write from end of **notMemS0**	–5	4	–5	4	–5	4	1,7
TWrLWrH	Early write pulse width	**d**–4	**d**+7	**d**–4	**d**+7	**d**–4	**d**+4	1,5
	Late write pulse width	**e**–4	**e**+7	**e**–4	**e**+7	**e**–4	**e**+4	1,6
All timings in nanoseconds (ns).								

Notes

1 Timing is for all write strobes **notMemWrB0-3**.

2 **a** is **T6** where **T6** can be from one to five periods **Tm** in length.

3 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

4 **c** is total of **T2**+**T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

5 **d** is total of **T3**+**T4**+**Twait**+**T5** where **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

6 **e** is total of **T4**+**Twait**+**T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

7 Sampled, not 100% tested.

Table 5.5 Write

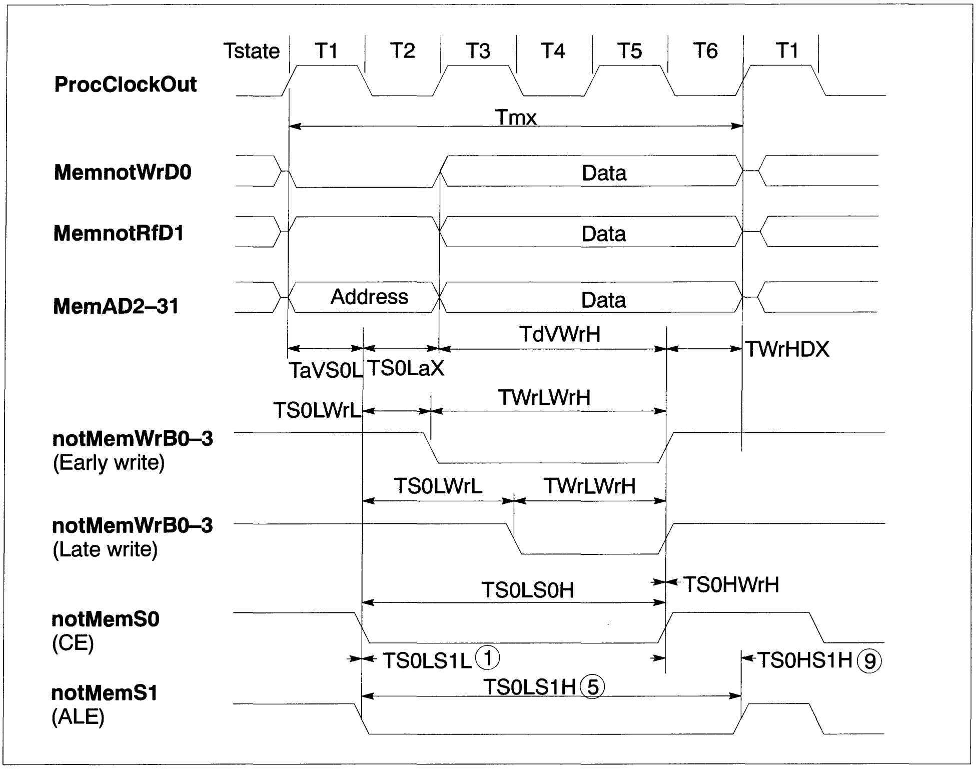


Figure 5.8 IMS T425 external write cycle

5.4 Wait

Taking **MemWait** high with the timing shown (figure 6.9) will extend the duration of **T4**. **MemWait** is sampled close to the falling edge of **ProcClockOut** prior to, but not at, the end of **T4**. By convention, **notMemS4** is used to synchronize wait state insertion. If this or another strobe is used, its delay should be such as to take the strobe low an even number of periods **Tm** after the start of **T1**, to coincide with a rising edge of **ProcClockOut**.

MemWait may be kept high indefinitely, although if dynamic memory refresh is used it should not be kept high long enough to interfere with refresh timing. **MemWait** operates normally during all cycles, including refresh and configuration cycles. It does not affect internal memory access in any way.

If the start of **T5** would coincide with a falling edge of **ProcClockOut** an extra wait period **Tm** (**EW**) is generated by the EMI to force coincidence with a rising edge. Rising edge coincidence is only forced if wait states are added, otherwise coincidence with a falling edge is permitted.

Symbol	Parameter	T425–20		T425–25		T425–30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TPCLWtH	Wait setup	10		10		8		ns	1,2
TPCLWtL	Wait hold	8		8		6		ns	1,2
TWtLWtH	Delay before re-assertion of Wait	50		40		33		ns	3

Notes

1 **ProcClockOut** load should not exceed 50pf.

2 If wait period exceeds refresh interval, refresh cycles will be lost.

3 Guaranteed, but not tested.

Table 5.6 IMS T425 memory wait

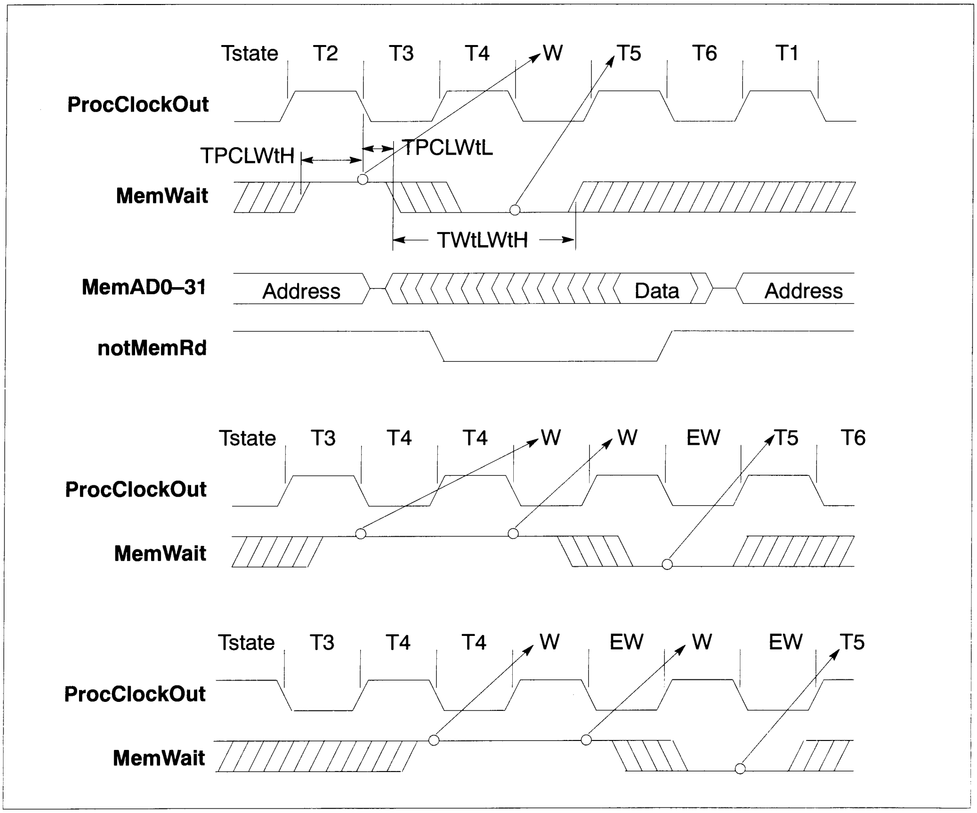


Figure 5.9 IMS T425 memory wait timing

5.5 Memory refresh

The **RefreshPending** pin is asserted high when the external memory interface is about to perform a refresh cycle. It remains high until the refresh cycle is started by the transputer. The minimum time for the **RefreshPending** pin to be high is for one cycle of **ProcClockOut** (two periods **Tm**), when the EMI was not about to perform a memory read or write. If the EMI was held in the tristate condition with **MemGranted** asserted, then **RefreshPending** will be asserted when the refresh controller in the EMI is ready to perform a refresh. **MemReq** may be re-asserted any time after the commencement of the refresh cycle. **RefreshPending** changes state near the rising edge of **ProcClockOut** and can therefore be sampled by the falling edge of **ProcClockOut**.

If no DMA is active then refresh will be performed following the end of the current internal or external memory cycle. If DMA is active the transputer will wait for DMA to terminate before commencing the refresh cycle. Unlike **MemnotRfD1**, **RefreshPending** is never tristated and can thus be interrogated by the DMA device; the DMA cycle can then be suspended, at the discretion of the DMA device, to allow refresh to take place.

The simple circuit of Figure 6.10 will suspend DMA requests from the external logic when **RefreshPending** is asserted, so that a memory refresh cycle can be performed. DMA is restored on completion of the refresh cycle. The transputer will not perform an external memory cycle other than a refresh cycle, using this method, until the requesting device removes its DMA request.

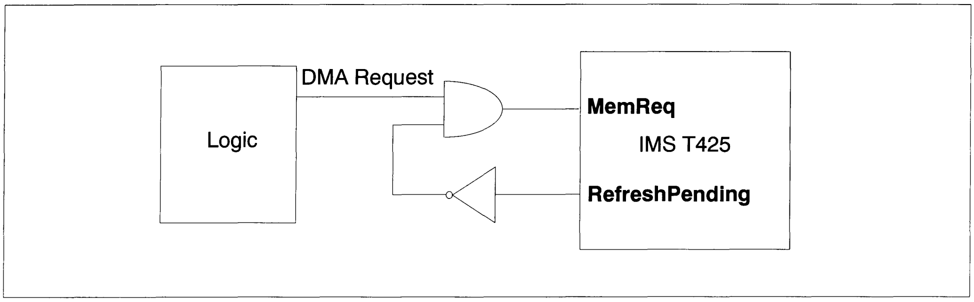


Figure 5.10 IMS T425 refresh with DMA

When refresh is disabled no refresh cycles occur. During the post-**Reset** period eight dummy refresh cycles will occur with the appropriate timing but with no bus or strobe activity.

A refresh cycle uses the same basic external memory timing as a normal external memory cycle, except that it starts two periods **Tm** before the start of **T1**. If a refresh cycle is due during an external memory access, it will be delayed until the end of that external cycle. Two extra periods **Tm** (periods **R** in the diagram) will then be inserted between the end of **T6** of the external memory cycle and the start of **T1** of the refresh cycle itself. The refresh address and various external strobes become active approximately one period **Tm** before **T1**. Bus signals are active until the end of **T2**, whilst **notMemRf** remains active until the end of **T6**.

For a refresh cycle, **MemnotRfD1** goes low before **notMemRf** goes low and **MemnotWrD0** goes high with the same timing as **MemnotRfD1**. All the address lines share the same timing, but only **MemAD2-11** give the refresh address. **MemAD12-30** stay high during the address period, whilst **MemAD31** remains low. Refresh cycles generate strobes **notMemS0-4** with timing as for a normal external cycle, but **notMemRd** and **notMemWrB0-3** remain high. **MemWait** operates normally during refresh cycles.

Refresh cycles do not interrupt internal memory accesses, although the internal addresses cannot be reflected on the external bus during refresh.

Symbol	Parameter	T425-20		T425-25		T425-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TRfLRfH	Refresh pulse width low	**a**–2	**a**+9	**a**–2	**a**+9	**a**–2	**a**+9	ns	1
TRaVS0L	Refresh address setup before **notMemS0**	**b**–12		**b**–12		**b**–12		ns	
TRfLS0L	Refresh indicator setup before **notMemS0**	**b**–4	**b**+6	**b**–4	**b**+6	**b**–4	**b**+6	ns	2

Notes

1 **a** is total **Tmx**+**Tm**.

2 **b** is total **T1**+**Tm** where **T1** can be from one to four periods **Tm** in length.

Table 5.7 Memory refresh

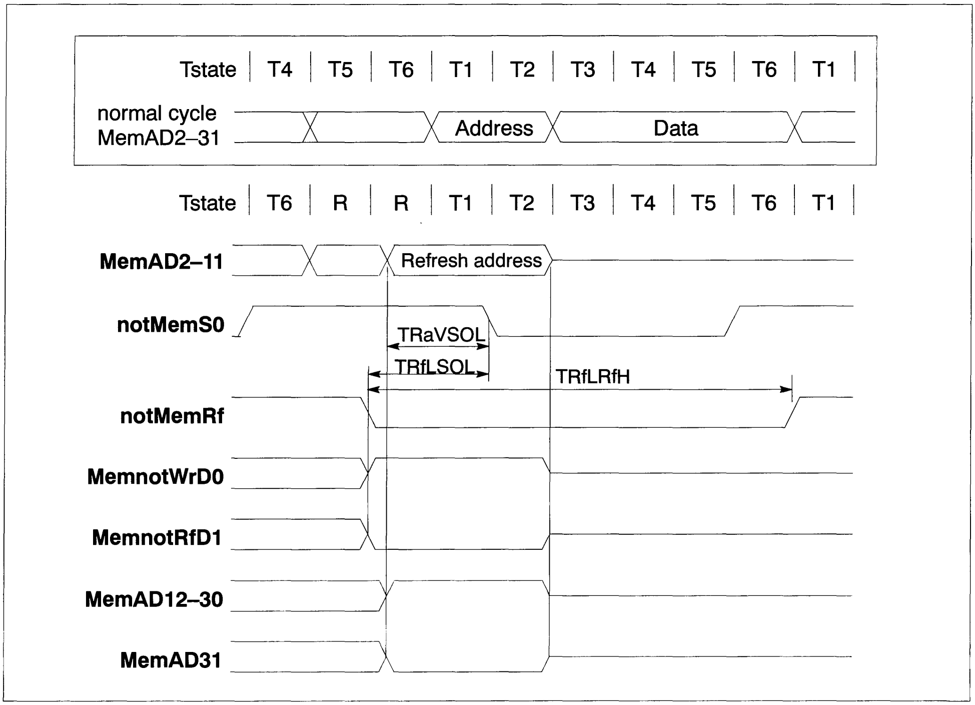


Figure 5.11 IMS T425 refresh cycle timing

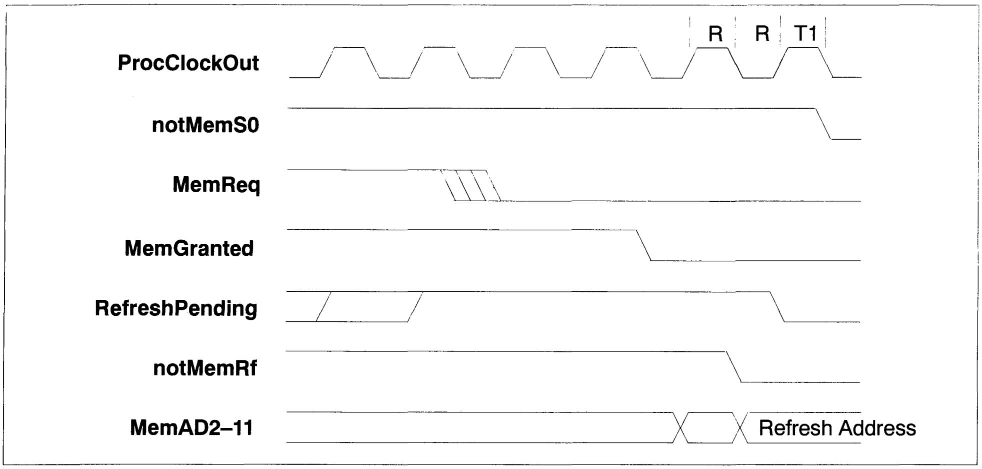


Figure 5.12 IMS T425 refresh pending timing diagram

5.6 Direct memory access

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high. The transputer samples **MemReq** just before falling edges of **ProcClockOut**. To guarantee taking over the bus immediately following either a refresh or external memory cycle, **MemReq** must be sampled at least four periods **Tm** before the end of **T6**. In the absence of an external memory cycle, the address bus is tristated two periods **Tm** after the **ProcClockOut** rising edge which follows the sample.

Removal of **MemReq** is sampled just before falling edges of **ProcClockOut** and **MemGranted** is removed synchronously with the second falling edge of **ProcClockOut** which follows the sample. If accurate timing of DMA is required, the setup time relative to **ProcClockOut** must be met. Further external bus activity, either refresh, external cycles or reflection of internal cycles, will commence at the next but one rising edge of **ProcClockOut**.

The strobes (**notMemS0–4** and **notMemWrB0–3**) are left in their inactive states during DMA. DMA cannot interrupt a refresh or external memory cycle, and outstanding refresh cycles will occur before the bus is released to DMA. DMA does not interfere with internal memory cycles in any way, although a program running in internal memory would have to wait for the end of DMA before accessing external memory. DMA cannot access internal memory. If DMA extends longer than one refresh interval (Memory Refresh Configuration Coding, table 6.11), the DMA user becomes responsible for refresh (see section 6.5). DMA may also inhibit an internally running program from accessing external memory.

DMA allows a bootstrap program to be loaded into external RAM ready for execution after reset. If **MemReq** is held high throughout reset, **MemGranted** will be asserted before the bootstrap sequence begins. **MemReq** must be high at least one period **TDCLDCL** of **ClockIn** before **Reset**. The circuit should be designed to ensure correct operation if **Reset** could interrupt a normal DMA cycle.

Symbol	Parameter	T425–20		T425–25		T425–30		Units	Note
		Min	Max	Min	Max	Min	Max		
TMRHPCL	**MemReq** setup before **ProcClock** falling	3	14	3	14	4	14	ns	1
TPCLMGH	**MemReq** response time	96	110	77	89	65	75	ns	2
TMRLPCL	**Memreq** removal before **ProcClock** falling	4	16	4	15	4	14		
TPCLMGL	**MemReq** end response time	50	66	40	54	37	49	ns	
TADZMGH	Bus tristate before **MemGranted**	0	27	0	22	0	18	ns	
TMGLADV	Bus active after end of **MemGranted**	0	32	0	26	0	20	ns	

Notes

1 Setup time need only be met to guarantee sampling on this edge.

2 If an external cycle is active, maximum time could be
(1 EMI cycle **Tmx**)+(1 refresh cycle **TRfLRfH**)+(6 periods **Tm**).

Table 5.8 Memory request

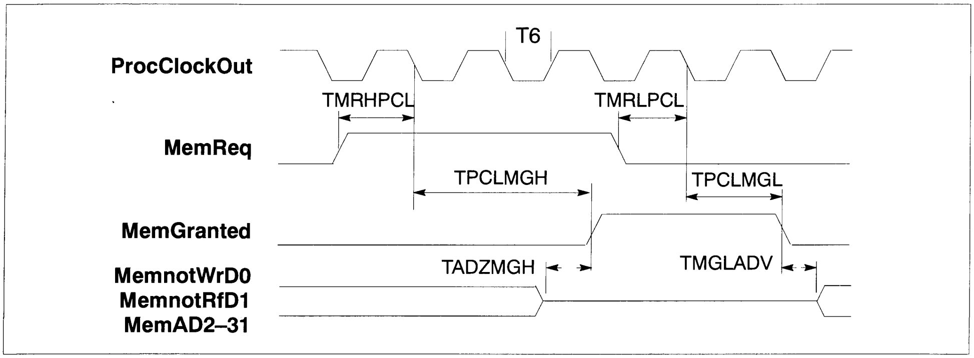
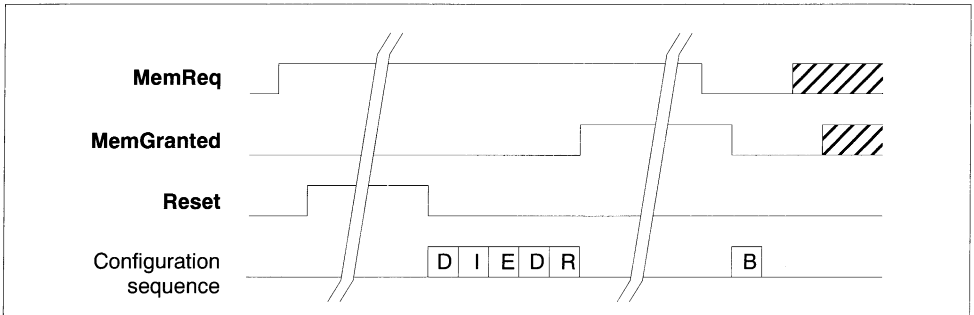


Figure 5.13 IMS T425 memory request timing



D Pre– and post–configuration delays (figure 4.3)
I Internal configuration sequence
E External configuration sequence
R Initial refresh sequence
B Bootstrap sequence

Figure 5.14 IMS T425 DMA sequence at reset

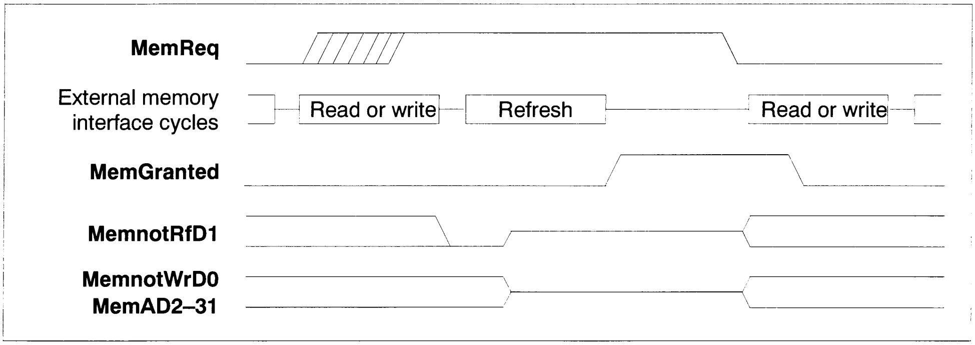


Figure 5.15 IMS T425 operation of **MemReq**, **MemGranted** with external, refresh memory cycles

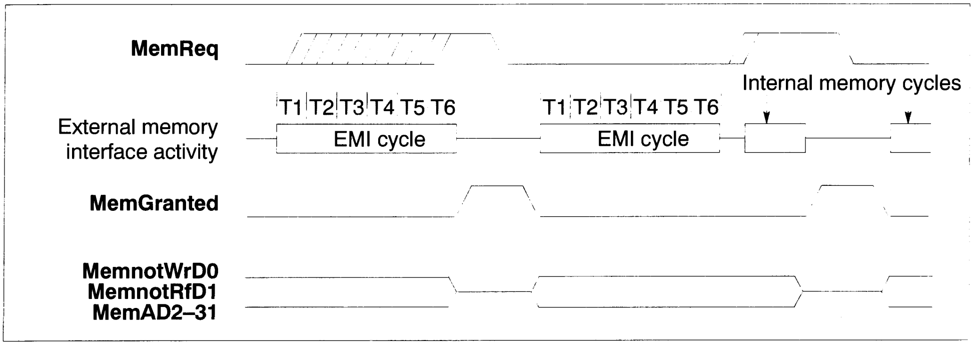


Figure 5.16 IMS T425 operation of **MemReq**, **MemGranted** with external, internal memory cycles

5.7 Memory configuration

MemConfig is an input pin used to read configuration data when setting external memory interface (EMI) characteristics. It is read by the processor on two occasions after **Reset** goes low; first to check if one of the preset internal configurations is required, then to determine a possible external configuration.

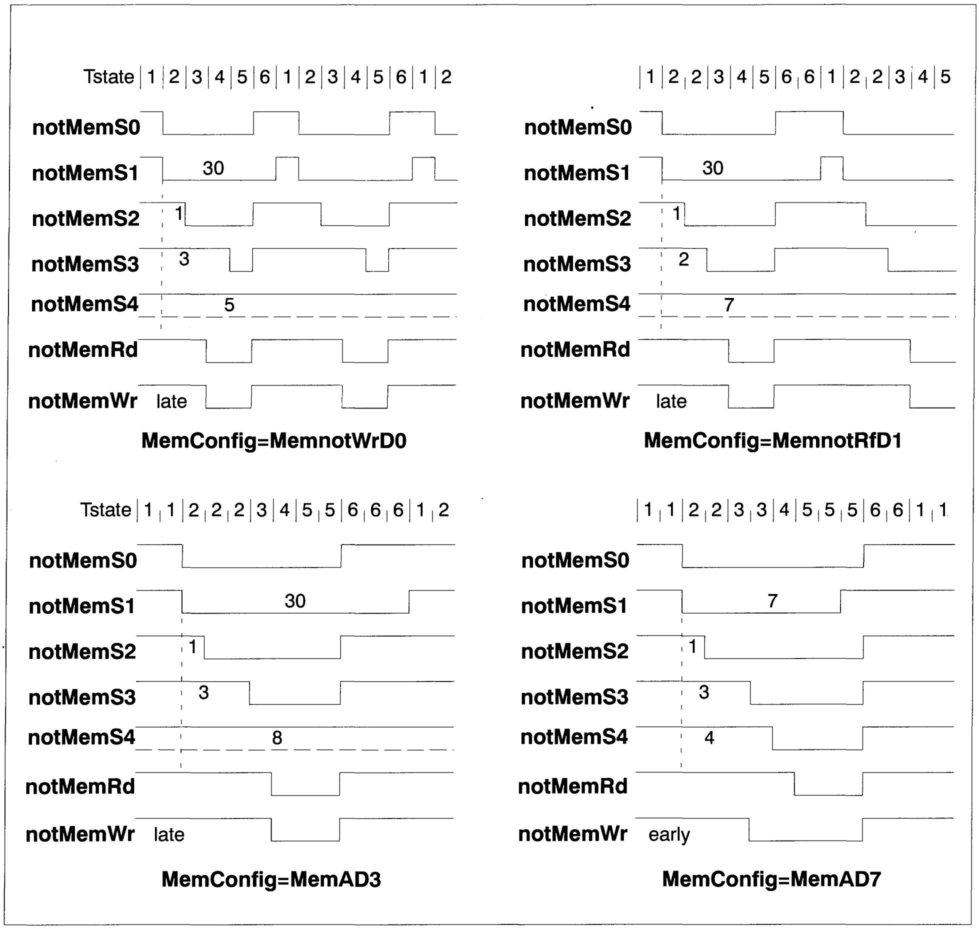
5.7.1 Internal configuration

The internal configuration scan comprises 64 periods **TDCLDCL** of **ClockIn** during the internal scan period of 144 **ClockIn** periods. **MemnotWrD0**, **MemnotRfD1** and **MemAD2-32** are all high at the beginning of the scan. Starting with **MemnotWrD0**, each of these lines goes low successively at intervals of two **ClockIn** periods and stays low until the end of the scan. If one of these lines is connected to **MemConfig** the preset internal configuration mode associated with that line will be used as the EMI configuration. The default configuration is that defined in the table for **MemAD31**; connecting **MemConfig** to **VDD** will also produce this default configuration. Note that only 17 of the possible configurations are valid, all others remain at the default configuration.

	Duration of each Tstate periods Tm						**Strobe coefficient**				**Write cycle**	**Refresh interval**	**Cycle time**
Pin	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**s1**	**s2**	**s3**	**s4**	**type**	**ClockIn cycles**	**Proc cycles**
MemnotWrD0	1	1	1	1	1	1	30	1	3	5	late	72	3
MemnotRfD1	1	2	1	1	1	2	30	1	2	7	late	72	4
MemAD2	1	2	1	1	2	3	30	1	2	7	late	72	5
MemAD3	2	3	1	1	2	3	30	1	3	8	late	72	6
MemAD4	1	1	1	1	1	1	3	1	2	3	early	72	3
MemAD5	1	1	2	1	2	1	5	1	2	3	early	72	4
MemAD6	2	1	2	1	3	1	6	1	2	3	early	72	5
MemAD7	2	2	2	1	3	2	7	1	3	4	early	72	6
MemAD8	1	1	1	1	1	1	30	1	2	3	early	†	3
MemAD9	1	1	2	1	2	1	30	2	5	9	early	†	4
MemAD10	2	2	2	2	4	2	30	2	3	8	late	72	7
MemAD11	3	3	3	3	3	3	30	2	4	13	late	72	9
MemAD12	1	1	2	1	2	1	4	1	2	3	early	72	4
MemAD13	2	1	2	1	2	2	5	1	2	3	early	72	5
MemAD14	2	2	2	1	3	2	6	1	3	4	early	72	6
MemAD15	2	1	2	3	3	3	8	1	2	3	early	72	7
MemAD31	4	4	4	4	4	4	31	30	30	18	late	72	12

† Provided for static RAM only.

Table 5.9 IMS T425 internal configuration coding



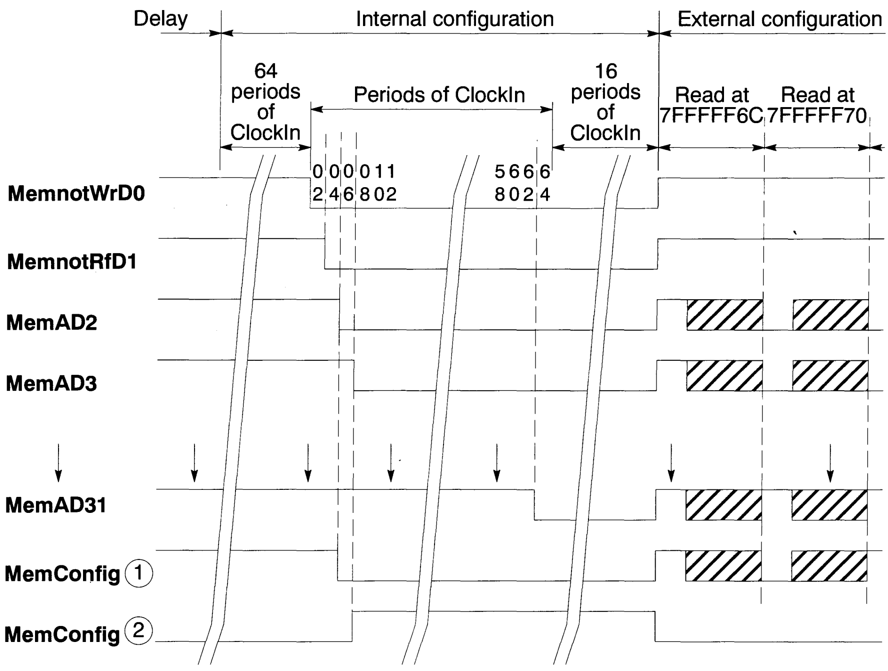
MemConfig=MemnotWrD0

MemConfig=MemnotRfD1

MemConfig=MemAD3

MemConfig=MemAD7

Figure 5.17 IMS T425 internal configuration



(1) Internal configuration: **MemConfig** connected to **MemAD2**

(2) External configuration: **MemConfig** connected to inverse of **MemAD3**

Figure 5.18 IMS T425 internal configuration scan

5.7.2 External configuration

If **MemConfig** is held low until **MemnotWrD0** goes low the internal configuration is ignored and an external configuration will be loaded instead. An external configuration scan always follows an internal one, but if an internal configuration occurs any external configuration is ignored.

The external configuration scan comprises 36 successive external read cycles, using the default EMI configuration preset by **MemAD31**. However, instead of data being read on the data bus as for a normal read cycle, only a single bit of data is read on **MemConfig** at each cycle. Addresses put out on the bus for each read cycle are shown in table 6.10, and are designed to address ROM at the top of the memory map. The table shows the data to be held in ROM; data required at the **MemConfig** pin is the inverse of this.

MemConfig is typically connected via an inverter to **MemnotWrD0**. Data bit zero of the least significant byte of each ROM word then provides the configuration data stream. By switching **MemConfig** between various data bus lines up to 32 configurations can be stored in ROM, one per bit of the data bus. **MemConfig** can be permanently connected to a data line or to **GND**. Connecting **MemConfig** to **GND** gives all **Tstates** configured to four periods; **notMemS1** pulse of maximum duration; **notMemS2-4** delayed by maximum; refresh interval 72 periods of **ClockIn**; refresh enabled; late write.

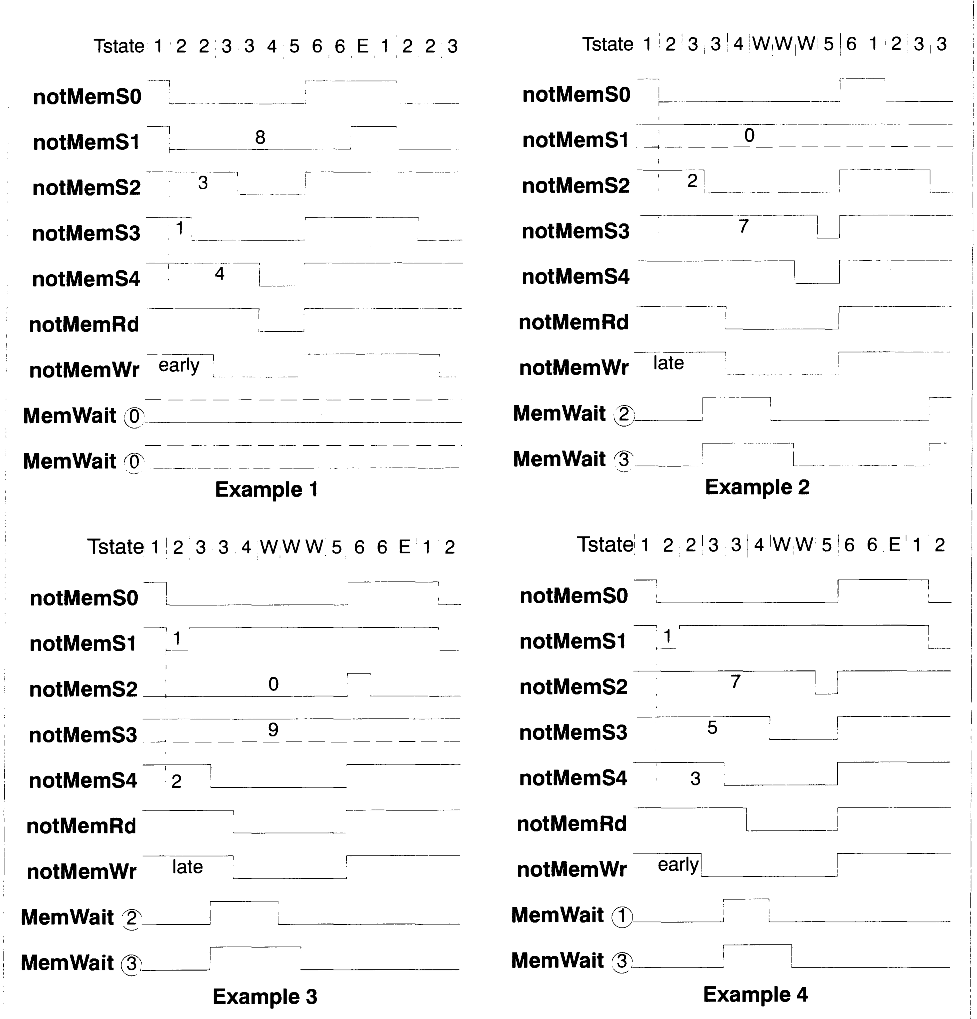
The external memory configuration table 6.10 shows the contribution of each memory address to the 13 configuration fields. The lowest 12 words (#7FFFFF6C to #7FFFFF98, fields 1 to 6) define the number of extra periods **Tm** to be added to each **Tstate**. If field 2 is 3 then three extra periods will be added to **T2** to extend it to the maximum of four periods.

The next five addresses (field 7) define the duration of **notMemS1** and the following fifteen (fields 8 to 10) define the delays before strobes **notMemS2-4** become active. The five bits allocated to each strobe allow durations of from 0 to 31 periods **Tm**, as described in strobes page 87.

Addresses #7FFFFFEC to #7FFFFFF4 (fields 11 and 12) define the refresh interval and whether refresh is to be used, whilst the final address (field 13) supplies a high bit to **MemConfig** if a late write cycle is required.

The columns to the right of the coding table show the values of each configuration bit for the four sample external configuration diagrams. Note the inclusion of period **E** at the end of **T6** in some diagrams. This is inserted to bring the start of the next **Tstate T1** to coincide with a rising edge of **ProcClockOut** (page 88).

Wait states **W** have been added to show the effect of them on strobe timing; they are not part of a configuration. In each case which includes wait states, two wait periods are defined. This shows that if a wait state would cause the start of **T5** to coincide with a falling edge of **ProcClockOut**, another period **Tm** is generated by the EMI to force it to coincide with a rising edge of **ProcClockOut**. This coincidence is only necessary if wait states are added, otherwise coincidence with a falling edge is permitted. Any configuration memory access is only permitted to be extended using wait, up to a total of 14 **ClockIn** periods.



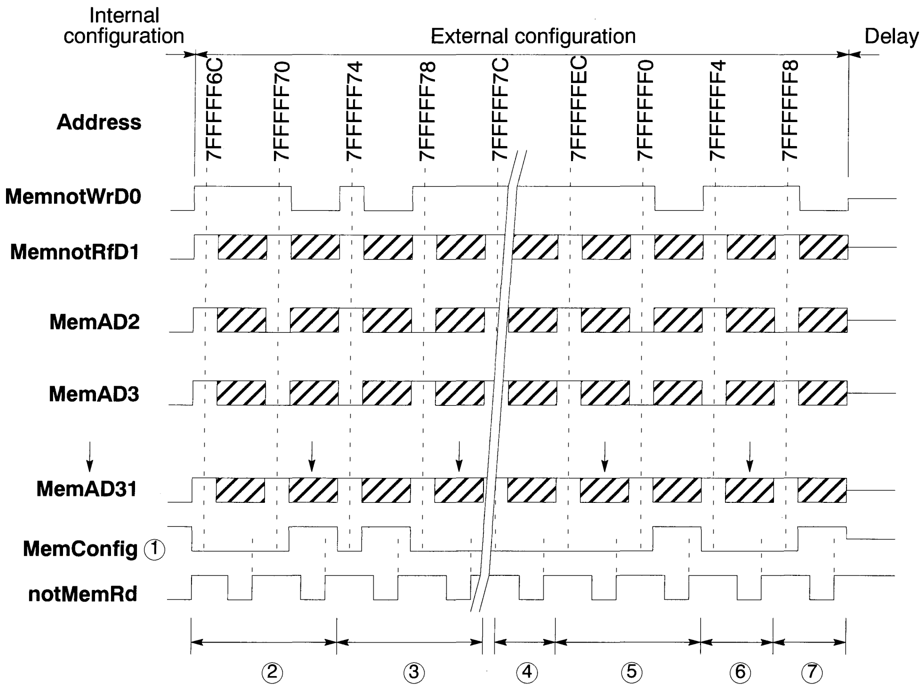
⓪ No wait states inserted

① One wait state inserted

② Two wait states inserted

③ Three wait states inserted

Figure 5.19 IMS T425 external configuration



① **MemConfig** connected to inverse of **MemnotWrD0**

② Configuration field 1; **T1** configured for 2 periods **Tm**

③ Configuration field 2; **T2** configured for 2 periods **Tm**

④ Configuration field 10; most significant bit of **notMemS4** configured high

⑤ Configuration field 11; refresh interval configured for 36 periods **ClockIn**

⑥ Configuration field 12; refresh enabled

⑦ Configuration field 13; early write cycle

Figure 5.20 IMS T425 external configuration scan

Scan cycle	Mem AD address	Field	Function	Example diagram			
				1	2	3	4
1	7FFFFF6C	1	T1 least significant bit	0	0	0	0
2	7FFFFF70	1	T1 most significant bit	0	0	0	0
3	7FFFFF74	2	T2 least significant bit	1	0	0	1
4	7FFFFF78	2	T2 most significant bit	0	0	0	0
5	7FFFFF7C	3	T3 least significant bit	1	1	1	1
6	7FFFFF80	3	T3 most significant bit	0	0	0	0
7	7FFFFF84	4	T4 least significant bit	0	0	0	0
8	7FFFFF88	4	T4 most significant bit	0	0	0	0
9	7FFFFF8C	5	T5 least significant bit	0	0	0	0
10	7FFFFF90	5	T5 most significant bit	0	0	0	0
11	7FFFFF94	6	T6 least significant bit	1	0	1	1
12	7FFFFF98	6	T6 most significant bit	0	0	0	0
13	7FFFFF9C	7	notMemS1 least significant bit	0	0	1	1
14	7FFFFFA0	7		0	0	0	0
15	7FFFFFA4	7	⇓ ⇓	0	0	0	0
16	7FFFFFA8	7		1	0	0	0
17	7FFFFFAC	7	notMemS1 most significant bit	0	0	0	0
18	7FFFFFB0	8	notMemS2 least significant bit	1	0	0	1
19	7FFFFFB4	8		1	1	0	1
20	7FFFFFB8	8	⇓ ⇓	0	0	0	1
21	7FFFFFBC	8		0	0	0	0
22	7FFFFFC0	8	notMemS2 most significant bit	0	0	0	0
23	7FFFFFC4	9	notMemS3 least significant bit	1	1	1	1
24	7FFFFFC8	9		0	1	0	0
25	7FFFFFCC	9	⇓ ⇓	0	1	0	1
26	7FFFFFD0	9		0	0	1	0
27	7FFFFFD4	9	notMemS3 most significant bit	0	0	0	0
28	7FFFFFD8	10	notMemS4 least significant bit	0	0	0	1
29	7FFFFFDC	10		0	1	1	1
30	7FFFFFE0	10	⇓ ⇓	1	1	0	0
31	7FFFFFE4	10		0	0	0	0
32	7FFFFFE8	10	notMemS4 most significant bit	0	0	0	0
33	7FFFFFEC	11	Refresh Interval least significant bit	-	-	-	-
34	7FFFFFF0	11	Refresh Interval most significant bit	-	-	-	-
35	7FFFFFF4	12	Refresh Enable	-	-	-	-
36	7FFFFFF8	13	Late Write	0	1	1	0

Table 5.10 IMS T425 external configuration coding

Refresh interval	Interval in µs	Field 11 encoding	Complete cycle (ms)
18	3.6	00	0.922
36	7.2	01	1.843
54	10.8	10	2.765
72	14.4	11	3.686

Table 5.11 IMS T425 memory refresh configuration coding

Refresh intervals are in periods of **ClockIn** and **ClockIn** frequency is 5 MHz:

Interval = 18 * 200 = 3600 ns

Refresh interval is between successive incremental refresh addresses.
Complete cycles are shown for 256 row DRAMS.

Symbol	Parameter	T425-20		T425-25		T425-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TMCVRdH	**MemConfig** data setup	25		20		15		ns	
TRdHMCX	**MemConfig** data hold	0		0		0		ns	
TS0LRdH	**notMemS0** to configuration data read	388	412	308	332	257	277	ns	

Table 5.12 Memory configuration

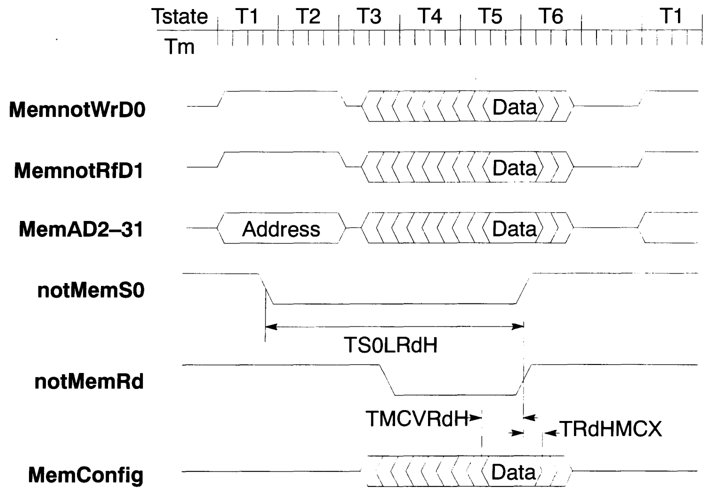


Figure 5.21 IMS T425 external configuration read cycle timing

6 Events

EventReq and **EventAck** provide an asynchronous handshake interface between an external event and an internal process. When an external event takes **EventReq** high the external event channel (additional to the external link channels) is made ready to communicate with a process. When both the event channel and the process are ready the processor takes **EventAck** high and the process, if waiting, is scheduled. **EventAck** is removed after **EventReq** goes low.

EventWaiting is asserted high by the transputer when a process executes an input on the event channel; typically with the occam `EVENT ? ANY` instruction. It remains high whilst the transputer is waiting for or servicing **EventReq** and is returned low when **EventAck** goes high. The **EventWaiting** pin changes near the falling edge of **ProcClockOut** and can therefore be sampled by the rising edge of **ProcClockOut**.

The **EventWaiting** pin can only be asserted by executing an *in* instruction on the event channel. The **EventWaiting** pin is not asserted high when an enable channel (*enbc*) instruction is executed on the Event channel (during an ALT construct in occam, for example). The **EventWaiting** pin can be asserted by executing the occam input on the event channel (such as `EVENT ? ANY`), provided that this does not occur as a guard in an alternative process. The **EventWaiting** pin can not be used to signify that an alternative process (ALT) is waiting on an input from the event channel.

EventWaiting allows a process to control external logic; for example, to clock a number of inputs into a memory mapped data latch so that the event request type can be determined.

Only one process may use the event channel at any given time. If no process requires an event to occur **EventAck** will never be taken high. Although **EventReq** triggers the channel on a transition from low to high, it must not be removed before **EventAck** is high. **EventReq** should be low during **Reset**; if not it will be ignored until it has gone low and returned high. **EventAck** is taken low when **Reset** occurs.

If the process is a high priority one and no other high priority process is running, the latency is as described on page 70. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

Cycle 1 Sample **EventReq** at pad on the rising edge of **ProcClockOut** and synchronize.

Cycle 2 Edge detect the synchronized **EventReq** and form the interrupt request.

Cycle 3 Sample interrupt vector for microcode ROM in the CPU.

Cycle 4 Execute the interrupt routine for the event rather than the next instruction.

		T425-20		**T425-25**		**T425-30**			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TVHKH	**EventReq** response	0		0		0		ns	1
TKHVL	**EventReq** hold	0		0		0		ns	1
TVLKL	Delay before removal of **EventAck**	0	157	0	127	0	107	ns	
TKLVH	Delay before re-assertion of **EventReq**	0		0		0		ns	1
TKHEWL	**EventAck** to end of **EventWaiting**	0		0		0		ns	1

Notes

1 Guaranteed, but not tested.

Table 6.1 Event timing

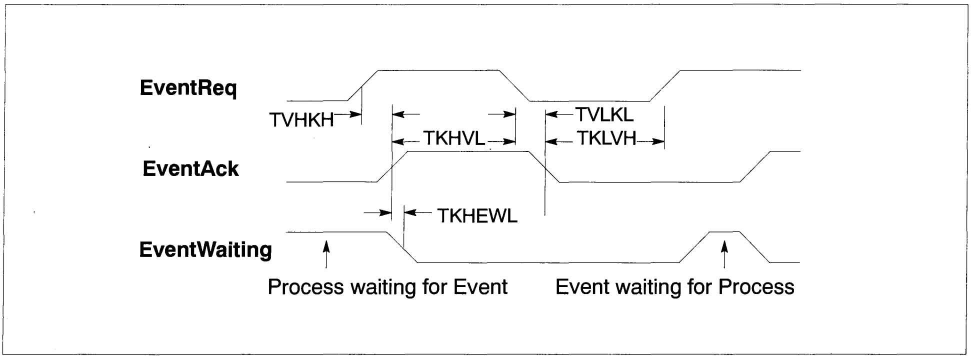


Figure 6.1 IMS T425 event timing

7 Links

Four identical INMOS bi-directional serial links provide synchronized communication between processors and with the outside world. Each link comprises an input channel and output channel. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

The IMS T425 links allow an acknowledge packet to be sent before the data packet has been fully received. This overlapped acknowledge technique is fully compatible with all other INMOS transputer links.

The IMS T425 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 5 or 20 Mbits/sec. Links are not synchronized with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimeters. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**, see figure 7.5. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions, see figure 7.6 . If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

Link speeds can be set by **LinkSpecial**, **Link0Special** and **Link123Special**. The link 0 speed can be set independently. Table 7.1 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed and as **Link123Special** for the others. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

Link	Linkn		Kbytes/sec	
Special	Special	Mbits/sec	Uni	Bi
0	0	10	910	1250
0	1	5	450	670
1	0	10	910	1250
1	1	20	1740	2350

Table 7.1 Speed settings for transputer links

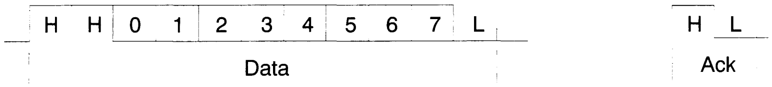


Figure 7.1 IMS T425 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	1
TJDf	**LinkIn** fall time				20	ns	1
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
		5 Mbits/s			30	ns	2
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		ohms	

Notes

1 Guaranteed, but not tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 7.2 Link timing

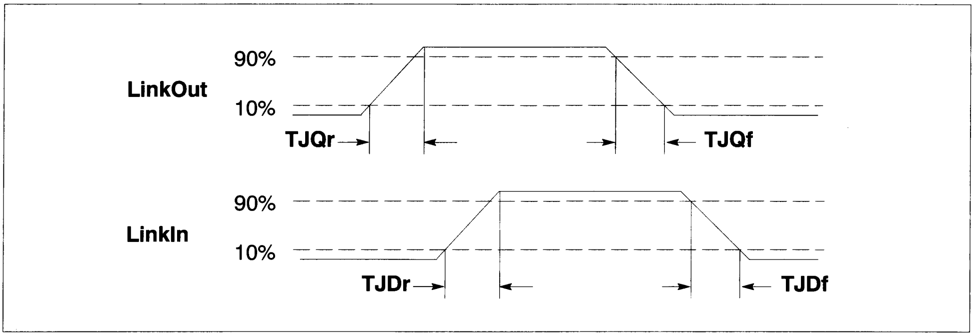


Figure 7.2 IMS T425 link timing

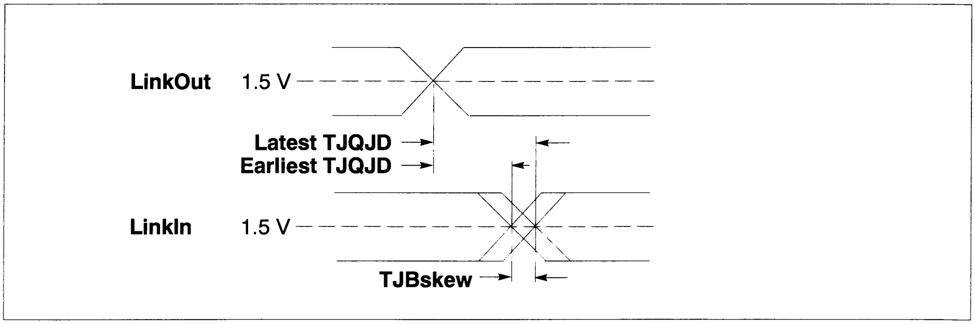


Figure 7.3 IMS T425 buffered link timing

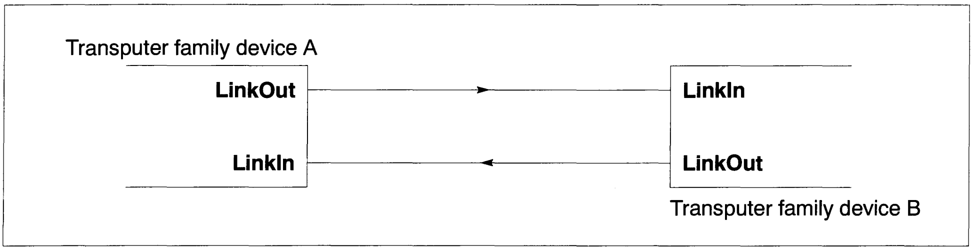


Figure 7.4 IMS T425 links directly connected

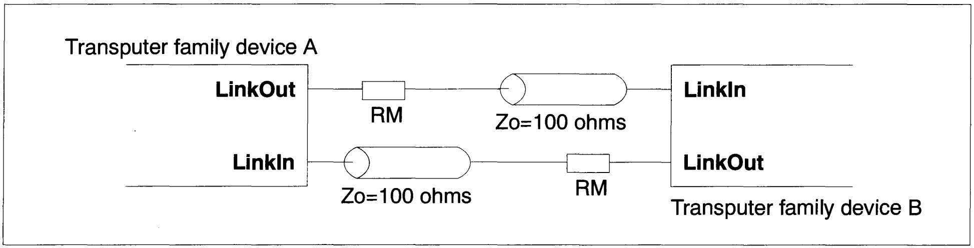


Figure 7.5 IMS T425 links connected by transmission line

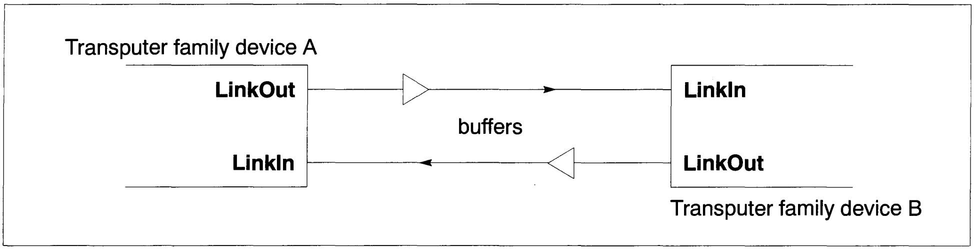


Figure 7.6 IMS T425 links connected by buffers

8 Electrical specifications

8.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
V_I, V_O	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
I_I	Input current		±25	mA	4
t_{OSC}	Output short circuit time (one pin)		1	s	2
T_S	Storage temperature	–65	150	°C	2
T_A	Ambient temperature under bias	–55	125	°C	2
P_{Dmax}	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 8.1 Absolute maximum ratings

8.2 Operating conditions

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
V_I, V_O	Input or output voltage	0	VDD	V	1,2
C_L	Load capacitance on any pin		60	pF	3
T_A	Operating temperature range	0	70	°C	4

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Excluding **LinkOut** load capacitance.

4 Air flow rate 400 linear ft/min transverse air flow.

Table 8.2 Operating conditions

8.3 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VIH	High level input voltage	2.0	VDD+0.5	V	1,2
VIL	Low level input voltage	–0.5	0.8	V	1,2
II	Input current @ GND<VI<VDD		±10	µA	1,2
VOH	Output high voltage @ IOH=2mA	VDD–1		V	1,2
VOL	Output low voltage @ IOL=4mA		0.4	V	1,2
IOZ	Tristate output current @ GND<V0<VDD		±10	µA	1,2
PD	Power dissipation		1	W	2,3
CIN	Input capacitance @ f=1MHz		7	pF	4
COZ	Output capacitance @ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

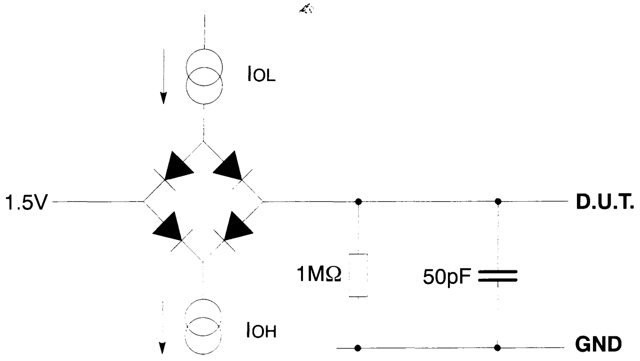
2 Parameters for IMS T425-S measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C.
Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution.
Power dissipation for processor operating at 20 MHz.

4 This parameter is sampled and not 100% tested.

Table 8.3 DC characteristics

8.4 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 8.1 Load circuit for AC measurements

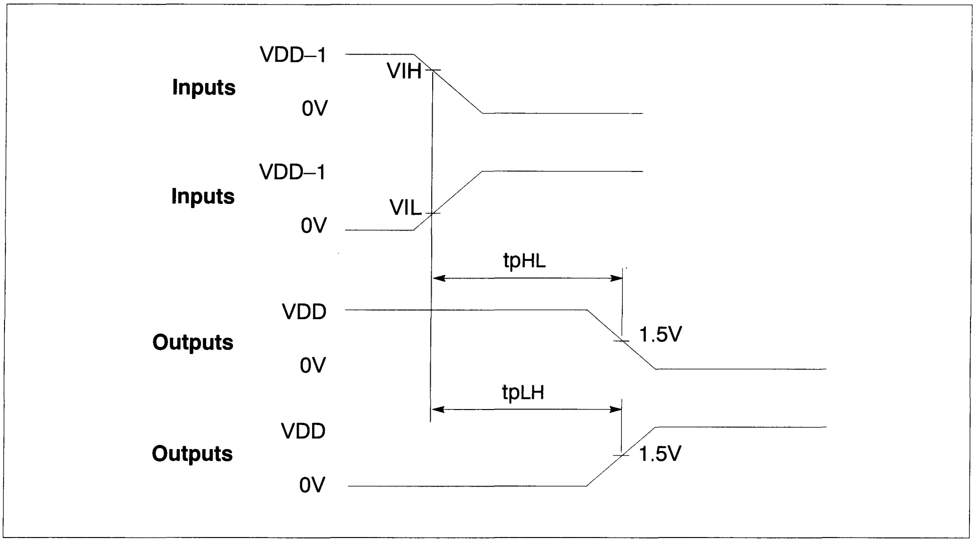


Figure 8.2 AC measurements timing waveforms

8.5 AC timing characteristics

Symbol	Parameter	Min	Max	Units	Notes
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3
TQr	Output rising edges		25	ns	1,3
TQf	Output falling edges		15	ns	1,3
TS0LaX	Address hold after **notMemS0**	**a**–8	**a**+8	ns	4

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 Guaranteed, but not tested.

4 **a** is **T2** where **T2** can be from one to four periods **Tm** in length.
Address lines include **MemnotWrD0**, **MemnotRfD1**, **MemAD2-31**.

Table 8.4 Input and output edges

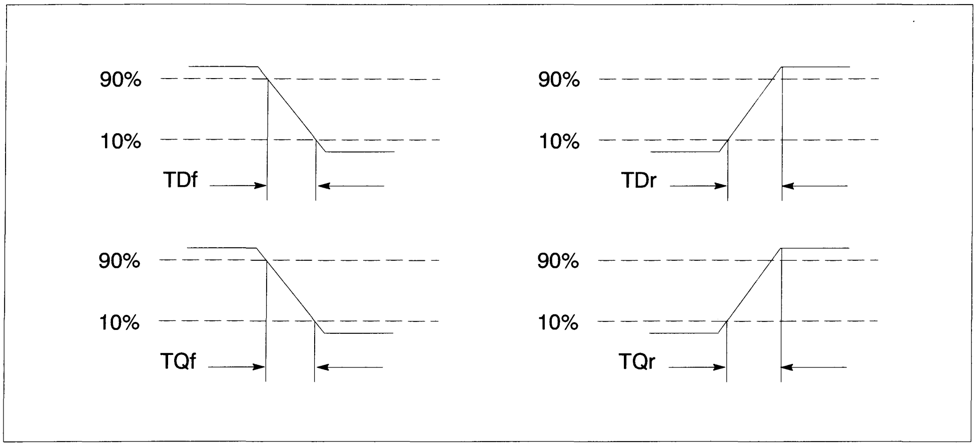


Figure 8.3 IMS T425 input and output edge timing

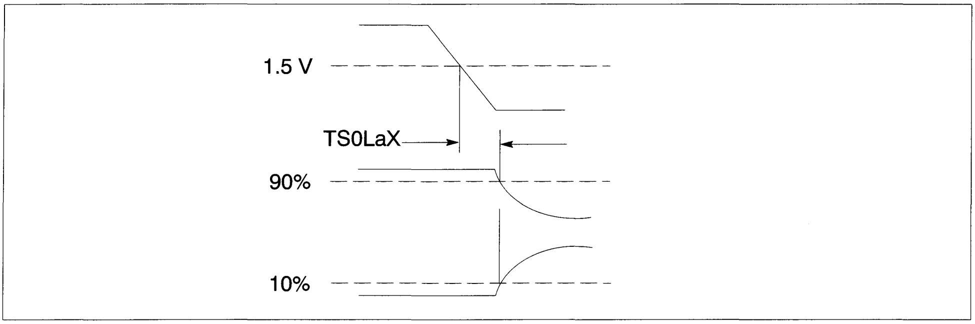
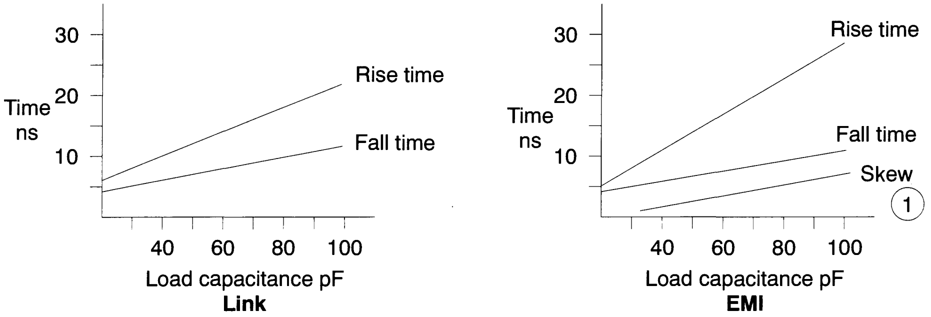


Figure 8.4 IMS T425 tristate timing relative to **notMemS0**



Notes

1 Skew is measured between **ProcClockOut** with a load of 2 Schottky TTL inputs and 30pF and **notMemS0** with a load of 2 Schottky TTL inputs and varying capacitance.

Figure 8.5 Typical rise/fall times

8.6 Power rating

Internal power dissipation (P_{INT}) of transputer and peripheral chips depends on **VDD**, as shown in figure 8.6. P_{INT} is substantially independent of temperature.

Total power dissipation (P_D) of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

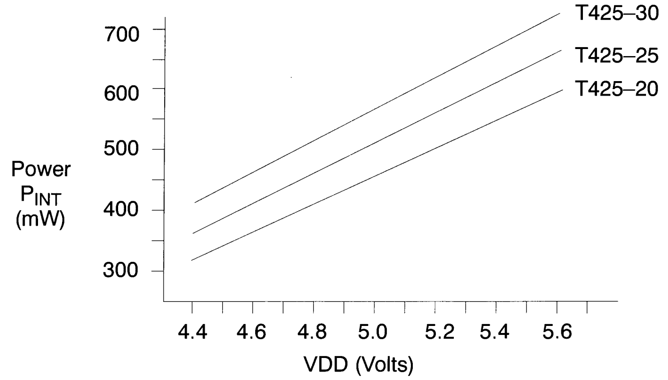
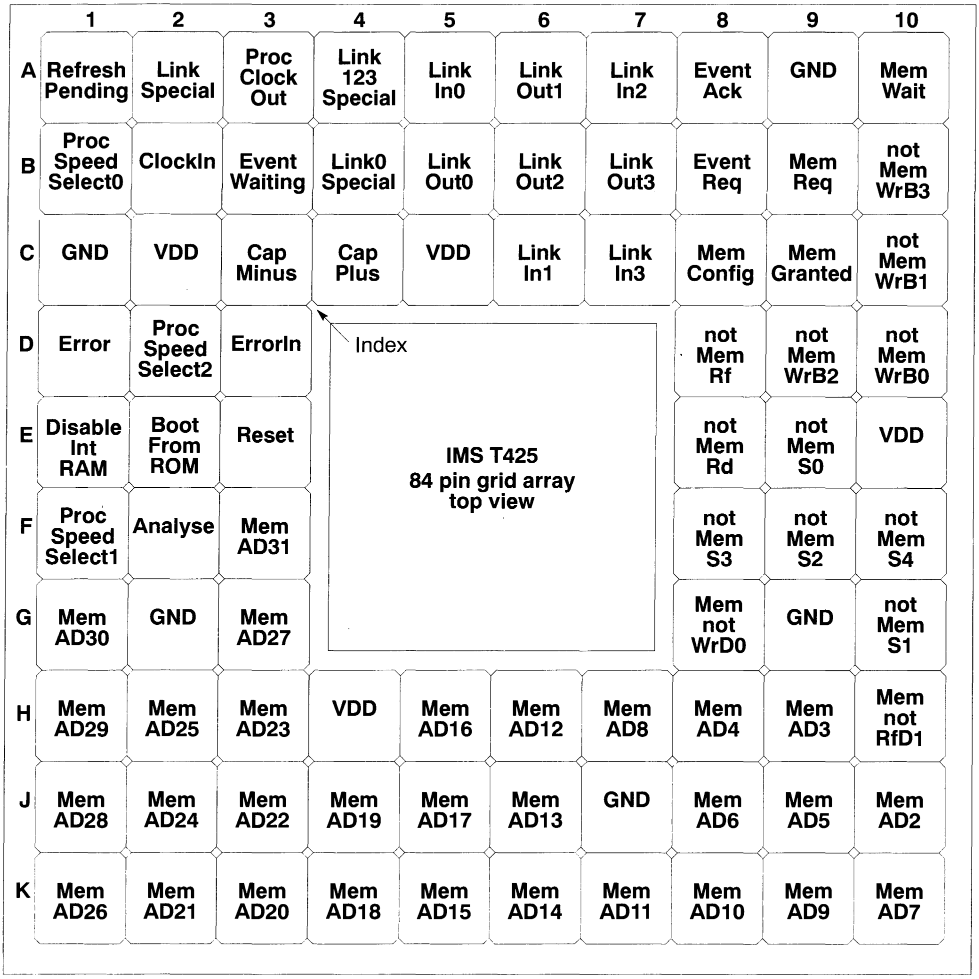


Figure 8.6 IMS T425 internal power dissipation vs VDD

9 Package pinouts

9.1 84 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Refresh Pending	Link Special	Proc Clock Out	Link 123 Special	Link In0	Link Out1	Link In2	Event Ack	GND	Mem Wait
B	Proc Speed Select0	ClockIn	Event Waiting	Link0 Special	Link Out0	Link Out2	Link Out3	Event Req	Mem Req	not Mem WrB3
C	GND	VDD	Cap Minus	Cap Plus	VDD	Link In1	Link In3	Mem Config	Mem Granted	not Mem WrB1
D	Error	Proc Speed Select2	ErrorIn					not Mem Rf	not Mem WrB2	not Mem WrB0
E	Disable Int RAM	Boot From ROM	Reset					not Mem Rd	not Mem S0	VDD
F	Proc Speed Select1	Analyse	Mem AD31					not Mem S3	not Mem S2	not Mem S4
G	Mem AD30	GND	Mem AD27					Mem not WrD0	GND	not Mem S1
H	Mem AD29	Mem AD25	Mem AD23	VDD	Mem AD16	Mem AD12	Mem AD8	Mem AD4	Mem AD3	Mem not RfD1
J	Mem AD28	Mem AD24	Mem AD22	Mem AD19	Mem AD17	Mem AD13	GND	Mem AD6	Mem AD5	Mem AD2
K	Mem AD26	Mem AD21	Mem AD20	Mem AD18	Mem AD15	Mem AD14	Mem AD11	Mem AD10	Mem AD9	Mem AD7

Figure 9.1 IMS T425 84 pin grid array package pinout

9.2 84 pin PLCC J-bend package

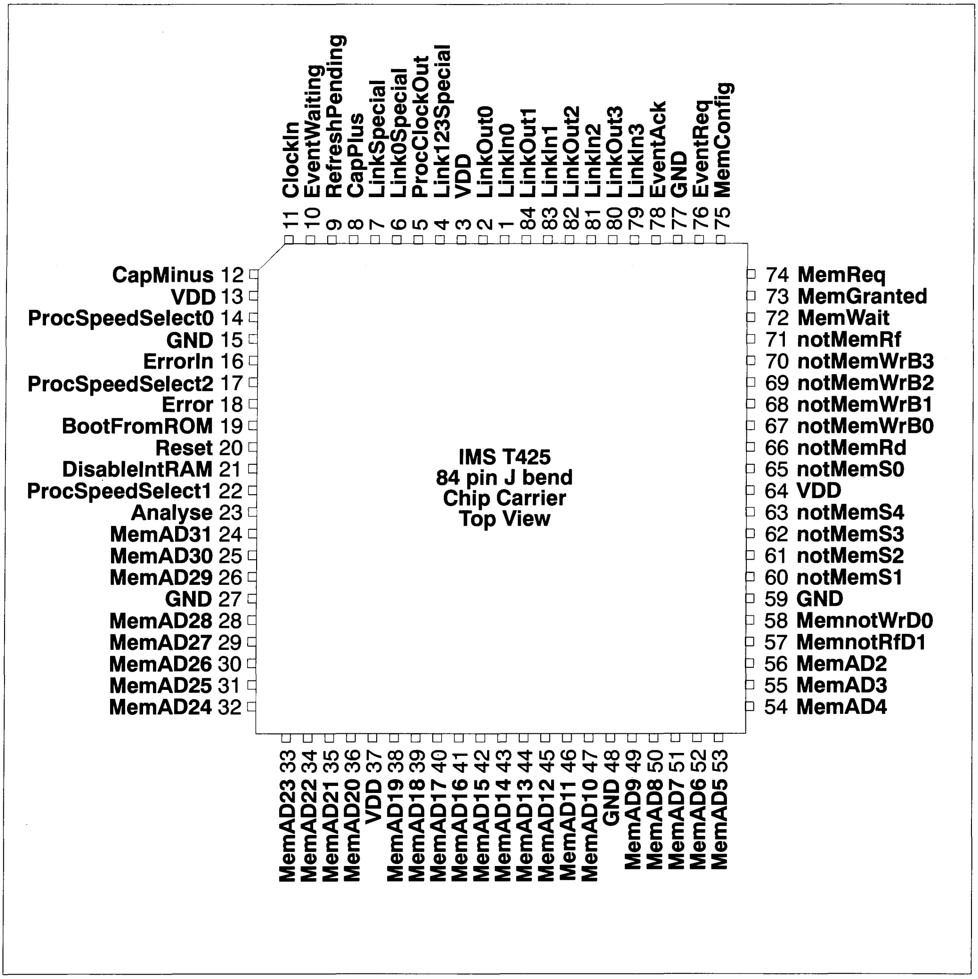
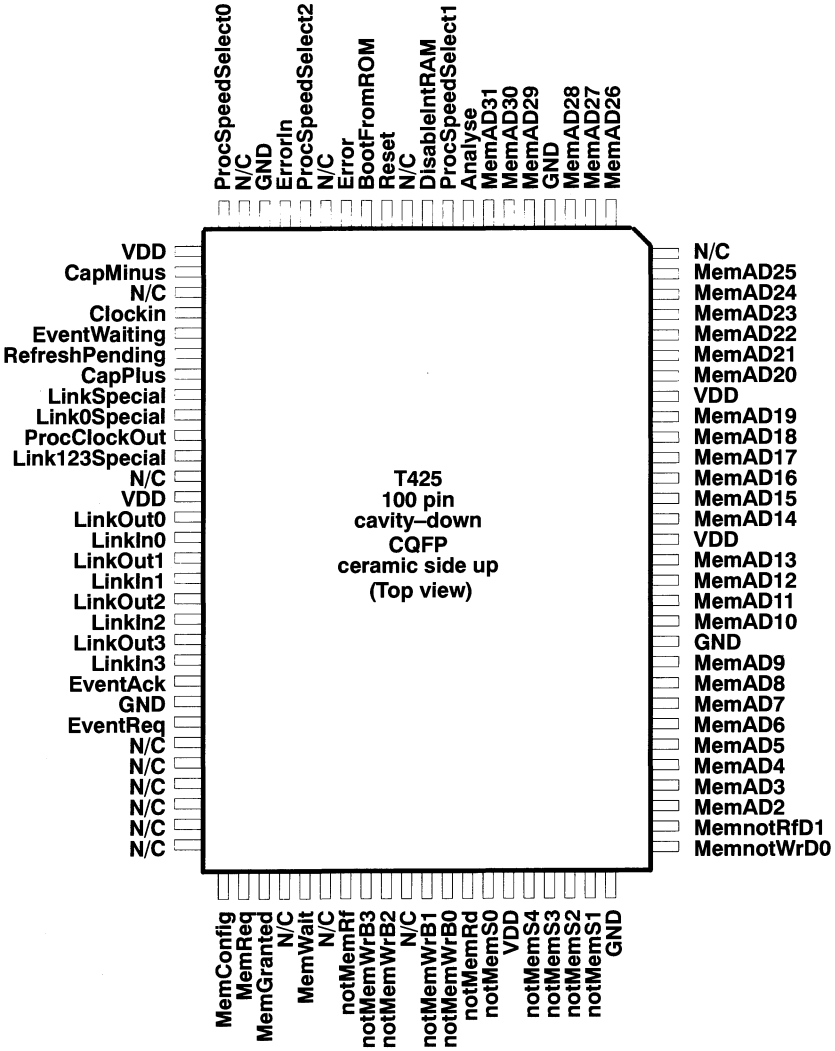


Figure 9.2 IMS T425 84 pin PLCC J-bend package pinout

9.3 100 pin cavity-down ceramic quad flat pack package



N/C indicates pins not connected

Figure 9.3 T425 100 pin cavity-down ceramic quad flat pack package pinout

10 Ordering

This section indicates the designation of speed and package selections for the various devices. Speed of **ClockIn** is 5 MHz for all parts. Transputer processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in the external memory section.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	**Processor clock speed**	**Processor cycle time**	**PLLx**	**Package**
IMS T425-G20S	20.0	50	4.0	84 pin ceramic pin grid array
IMS T425-G25S	25.0	40	5.0	84 pin ceramic pin grid array
IMS T425-G30S	30.0	33	6.0	84 pin ceramic pin grid array
IMS T425-J20S	20.0	50	4.0	84 pin plastic PLCC J–bend
IMS T425-J25S	25.0	40	5.0	84 pin plastic PLCC J–bend
IMS T425-F20S	20.0	50	4.0	100 pin ceramic quad flat pack
IMS T425-F25S	25.0	40	5.0	100 pin ceramic quad flat pack
IMS T425-F30S	30.0	33	6.0	100 pin ceramic quad flat pack

Table 10.1 IMS T425 ordering details

IMS T400 transputer

inmos®

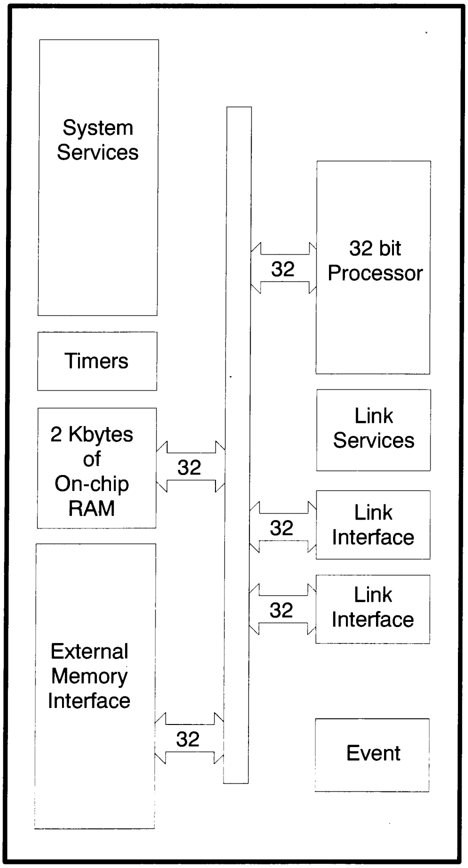
Engineering Data

FEATURES

32 bit architecture
50 ns internal cycle time
20 MHz only
20 MIPS peak instruction rate
10 MIPS sustained instruction rate
Pin compatible with IMS T805, IMS T800, IMS T425 and IMS T414
2 Kbytes on-chip static RAM
80 Mbytes/sec sustained data rate to internal memory
4 Gbytes directly addressable external memory
26 Mbytes/sec sustained data rate to external memory
950 ns response to interrupts
Whetstones/sec 704K
Dhrystones/sec 8193
Two INMOS serial links 5/10/20 Mbits/sec
High performance graphics support with block move instructions
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ±10% power supply
Packaging 84 pin PLCC/ 100 pin PQFP/ 84 pin PGA

APPLICATIONS

The IMS T400 is a low-cost product designed for single or multiprocessor applications in:
Telecoms
Office systems
Industrial control
Robotics
Instrumentation
Computing



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992

1 Introduction

The IMS T400 transputer is a 32 bit CMOS microcomputer with graphics support. It has 2 Kbytes on-chip RAM for high speed processing, a configurable memory interface and two standard INMOS communication links. The instruction set achieves efficient implementation of high level languages and provides direct support for the occam model of concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond.

For convenience of description, the IMS T400 operation is split into the basic blocks shown in figure 1.1.

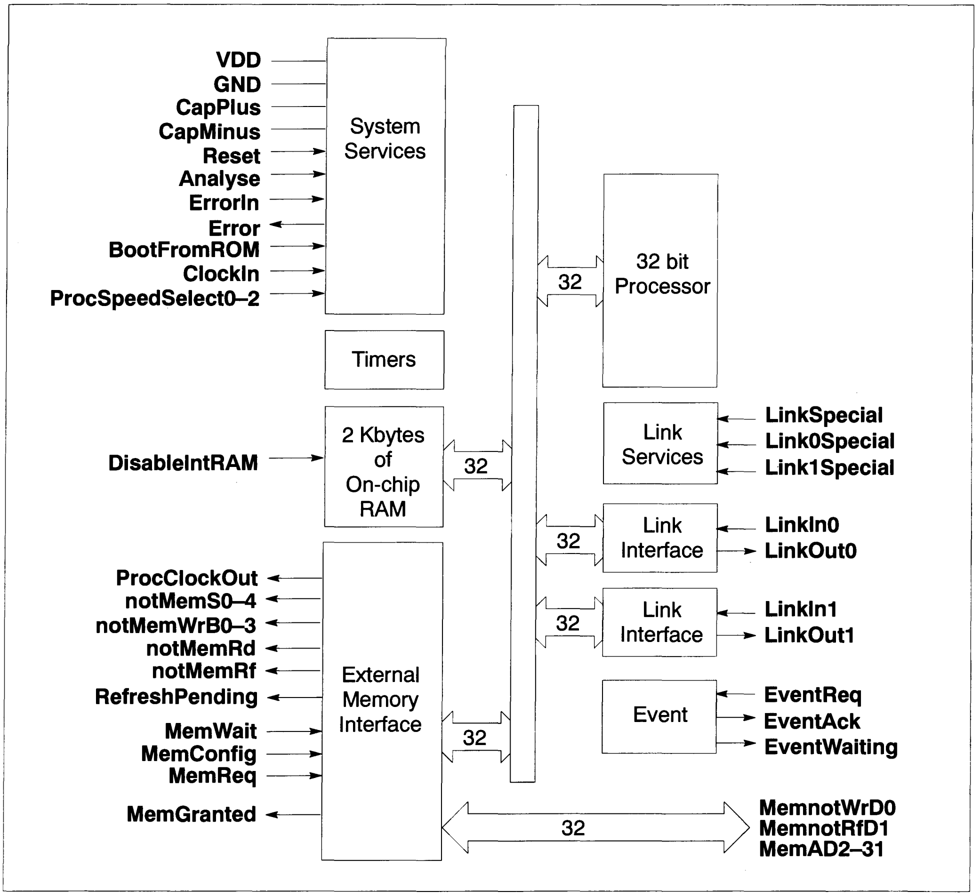


Figure 1.1 IMS T400 block diagram

The processor speed of the device is fixed at 20 MHz and achieves an instruction throughput of 20 MIPS peak and 10 MIPS sustained.

High performance graphics support is provided by microcoded block move instructions which operate at the speed of memory. The two-dimensional block move instructions provide for contiguous block moves as well as block copying of either non-zero bytes of data only or zero bytes only. Block move instructions can be used to provide graphics operations such as text manipulation, windowing, panning, scrolling and screen updating.

Cyclic redundancy checking (CRC) instructions are available for use on arbitrary length serial data streams, to provide error detection where data integrity is critical. Another feature of the IMS T400, useful for pattern recognition, is the facility to count bits set in a word.

The IMS T400 can directly access a linear address space of 4 Gbytes. The 32 bit wide memory interface uses multiplexed data and address lines and provides a data rate of up to 4 bytes every 150 nanoseconds (26.6 Mbytes/sec) for a 20 MHz device. A configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems.

System Services include processor reset and bootstrap control, together with facilities for error analysis. Error signals may be daisy-chained in multi-transputer systems.

The standard INMOS communication links allow networks of transputer family products to be constructed by direct point to point connections with no external logic. The IMS T400 links support the standard operating speed of 10 Mbits/sec, but also operate at 5 or 20 Mbits/sec. Each link can transfer data bi-directionally at up to 2.35 Mbytes/sec.

The IMS T400-20 is pin compatible with the IMS T805-20 and IMS T425-20 and can be plugged directly into a circuit designed for those devices.

The transputer is designed to implement the occam language, detailed in the occam *Reference Manual*, but also efficiently supports other languages such as C, Pascal and Fortran. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*. The instruction set of the IMS T400 is a superset of the IMS T800, excluding the FPU instructions. Additional instructions are listed in Chapter 4.

This datasheet supplies hardware implementation and characterisation details for the IMS T400. It is intended to be read in conjunction with the Transputer Architecture chapter, which details the architecture of the transputer and gives an overview of occam.

2 Pin designations

Signal names are preceded by **not** if they are active low, otherwise they are active high.
Pinout details for various packages are given on page 318.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
ProcSpeedSelect0-2	in	Processor speed selectors
Reset	in	System reset
Error	out	Error indicator
ErrorIn	in	Error daisychain input
Analyse	in	Error analysis
BootFromROM	in	Boot from external ROM or from link
DisableIntRAM	in	Disable internal RAM
HoldToGND		Must be connected to **GND**
DoNotWire		Must not be wired
N/C		There is no internal connection to this pin

Table 2.1 IMS T400 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemnotWrD0	in/out	Multiplexed data bit 0 and write cycle warning
MemnotRfD1	in/out	Multiplexed data bit 1 and refresh warning
MemAD2-31	in/out	Multiplexed data and address bus
notMemRd	out	Read strobe
notMemWrB0-3	out	Four byte-addressing write strobes
notMemS0-4	out	Five general purpose strobes
notMemRf	out	Dynamic memory refresh indicator
RefreshPending	out	Dynamic refresh is pending
MemWait	in	Memory cycle extender
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted
MemConfig	in	Memory configuration data input

Table 2.2 IMS T400 external memory interface

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge
EventWaiting	out	Event input requested by software

Table 2.3 IMS T400 event

Pin	In/Out	Function
LinkIn0-1	in	Two serial data input channels
LinkOut0-1	out	Two serial data output channels
LinkSpecial	in	Select non-standard speed as 5 or 20 Mbits/sec
Link0Special	in	Select special speed for Link 0
Link1Special	in	Select special speed for Link 1

Table 2.4 IMS T400 link

3 System services

System services include all the necessary logic to initialise and sustain operation of the device. They also include error handling and analysis facilities.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimise inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 10 MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

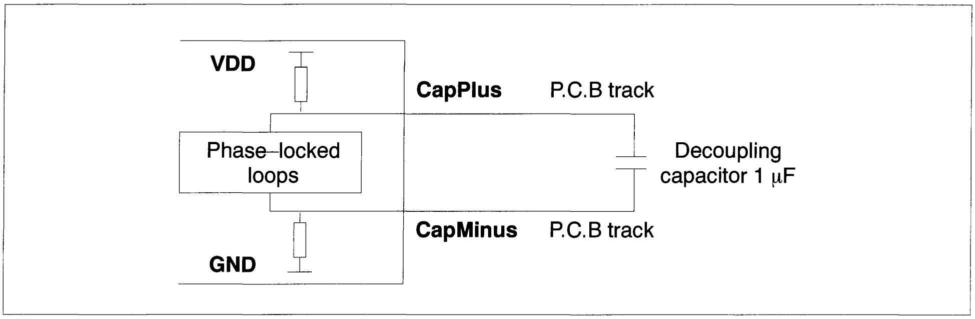


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	T400-20			Units	Notes
		Min	Nom	Max		
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1, 2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1, 3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1, 4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200 ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 8.3).

Table 3.1 Input clock

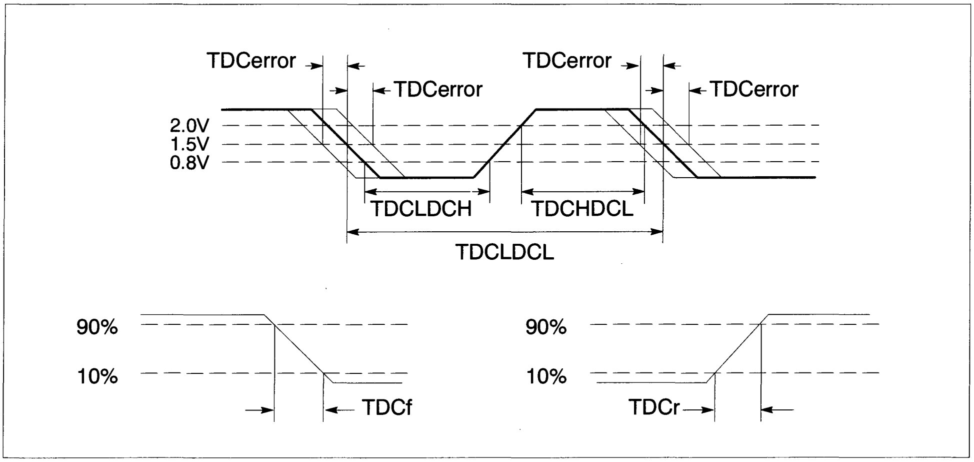


Figure 3.2 **ClockIn** timing

3.4 ProcSpeedSelect0–2

Processor speed of the IMS T400 is variable in discrete steps. The desired speed can be selected, up to the maximum rated for a particular component, by the three speed select lines **ProcSpeedSelect0-2**. The pins are tied high or low, according to table 3.2, for the various speeds.

Currently only the 20 MHz speed select combination is available on the IMS T400; the others are not valid speed selectors. The frequency of **ClockIn** for the speeds given in the table is 5 MHz.

ProcSpeed-Select2	ProcSpeed-Select1	ProcSpeed-Select0	Processor Clock Speed MHz	Processor Cycle Time ns	Notes
0	0	0	20.0	50.0	
0	0	1	22.5	44.4	Not supported
0	1	0	25.0	40.0	Not supported
0	1	1	30.0	33.3	Not supported
1	0	0	35.0	28.6	Not supported
1	0	1			Invalid
1	1	0	17.5	57.1	Not supported
1	1	1			Invalid

Table 3.2 Processor speed selection

3.5 Bootstrap

The transputer can be bootstrapped either from a link or from external ROM. To facilitate debugging, **BootFromROM** may be dynamically changed but must obey the specified timing restrictions. It is sampled once only by the transputer, before the first instruction is executed after **Reset** is taken low.

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFFFFFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the **W** register points to *MemStart* (page 4).

If **BootFromROM** is connected low (e.g. to **GND**) the transputer will wait for the first bootstrap message to arrive on any one of its links. The transputer is ready to receive the first byte on a link within two processor cycles **TPCLPCL** after **Reset** goes low.

If the first byte received (the control byte) is greater than 1 it is taken as the quantity of bytes to be input. The following bytes, to that quantity, are then placed in internal memory starting at location *MemStart*. Following reception of the last byte the transputer will start executing code at *MemStart* as a low priority process. **BootFromROM** may be taken high after reception of the last byte, if required. The memory space immediately above the loaded code is used as work space. A byte arriving on the other link after the control byte has been received and on the bootstrapping link after the last bootstrap byte, will be retained and no acknowledge will be sent until a process inputs from them.

3.6 Peek and poke

Any location in internal or external memory can be interrogated and altered when the transputer is waiting for a bootstrap from link. If the control byte is 0 then eight more bytes are expected on the same link. The first four byte word is taken as an internal or external memory address at which to poke (write) the second four byte word. If the control byte is 1 the next four bytes are used as the address from which to peek (read) a word of data; the word is sent down the output channel of the same link.

Following such a peek or poke, the transputer returns to its previously held state. Any number of accesses may be made in this way until the control byte is greater than 1, when the transputer will commence reading its bootstrap program. Either link can be used, but addresses and data must be transmitted via the same link as the control byte.

3.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initialises the transputer, triggers the memory configuration sequence and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) must not occur whilst **Reset** is high but can occur before bootstrap (page 298).

After the end of **Reset** there will be a delay of 144 periods of **ClockIn** (figure 3.3). Following this, the **MemnotWrD0**, **MemnotRfD1** and **MemAD2-31** pins will be scanned to check for the existence of a pre-programmed memory interface configuration (page 300). This lasts for a further 144 periods of **ClockIn**. Regardless of whether a configuration was found, 36 configuration read cycles will then be performed on external memory using the default memory configuration (page 302), in an attempt to access the external configuration ROM. A delay will then occur, its period depending on the actual configuration. Finally eight complete and consecutive refresh cycles will initialise any dynamic RAM, using the new memory configuration. If the memory configuration does not enable refresh of dynamic RAM the refresh cycles will be replaced by an equivalent delay with no external memory activity.

If **BootFromROM** is high bootstrapping will then take place immediately, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

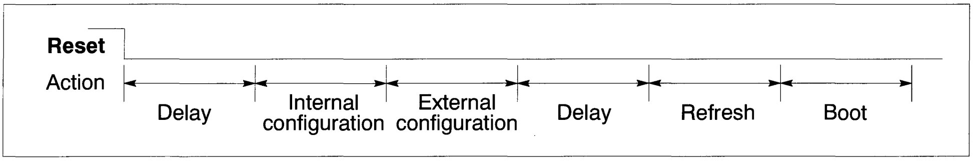


Figure 3.3 IMS T400 post–reset sequence

3.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 48). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine.

Processor flags **Error**, **HaltOnError** and **EnableJ0Break** are normally cleared at reset on the IMS T400; however, if **Analyse** is asserted the flags are not altered. Memory refresh continues.

Input links will continue with outstanding transfers. Output links will not make another access to memory for data but will transmit only those bytes already in the link buffer. Providing there is no delay in link acknowledgement, the links should be inactive within a few microseconds of the transputer halting.

Reset should not be asserted before the transputer has halted and link transfers have ceased. When **Reset** is taken low whilst **Analyse** is high, neither the memory configuration sequence nor the block of eight refresh cycles will occur; the previous memory configuration will be used for any external memory accesses. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 3.3.

I	**MemStart** if bootstrapping from a link, or the external memory bootstrap address if bootstrapping from ROM.
W	**MemStart** if bootstrapping from ROM, or the address of the first free word after the bootstrap program if bootstrapping from link.
A	The value of **I** when the processor halted.
B	The value of **W** when the processor halted, together with the priority of the process when the transputer was halted (i.e. the **W** descriptor).
C	The ID of the bootstrapping link if bootstrapping from link.

Table 3.3 Register values after Analyse

Symbol	Parameter	T400-20			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

1 Full periods of **ClockIn TDCLDCL** required.

2 At power-on reset.

3 Must be stable until after end of bootstrap period. See Bootstrap section 3.5.

Table 3.4 **Reset** , **Analyse** and **BootFromROM** timing

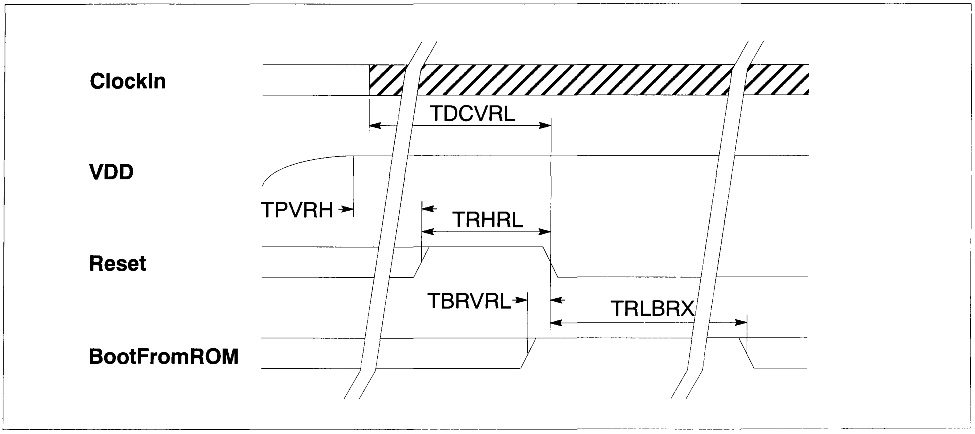


Figure 3.4 Transputer **Reset** timing with **Analyse** low

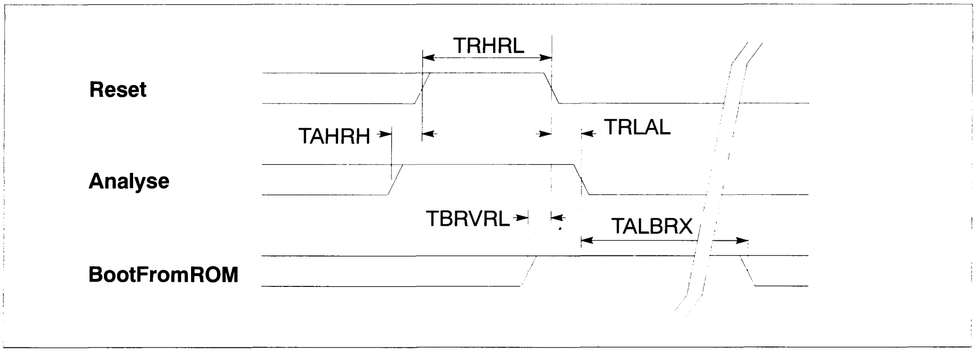


Figure 3.5 Transputer **Reset**, **Analyse** and **BootFromROM** timing

3.9 Error, ErrorIn

The **Error** pin carries the OR'ed output of the internal *Error* flag and the **ErrorIn** input. If **Error** is high it indicates either that **ErrorIn** is high or that an error was detected in one of the processes. An internal error can be caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly. Once set, the *Error* flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (page 278).

A process can be programmed to stop if the *Error* flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data. **ErrorIn** does not directly affect the status of a processor in any way.

By setting the *HaltOnError* flag the transputer itself can be programmed to halt if *Error* becomes set. If *Error* becomes set after *HaltOnError* has been set, all processes on that transputer will cease but will not necessarily cause other transputers in a network to halt. Setting *HaltOnError* after *Error* will not cause the transputer to halt; this allows the processor reset and analyse facilities to function with the flags in indeterminate states.

An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by 'daisy-chaining' the **ErrorIn** and **Error** pins of a number of processors and applying the final **Error** output signal to the **EventReq** pin of a suitably programmed master transputer. Since the process state is preserved when stopped by an error, the master transputer can then use the analyse function to debug the fault. When using such a circuit, note that the *Error* flag is in an indeterminate state on power up; the circuit and software should be designed with this in mind.

Error checks can be removed completely to optimise the performance of a proven program; any unexpected error then occurring will have an arbitrary undefined effect.

If a high priority process pre-empts a low priority one, status of the *Error* and *HaltOnError* flags is saved for the duration of the high priority process and restored at the conclusion of it. Status of both flags is transmitted to the high priority process. Either flag can be altered in the process without upsetting the error status of any complex operation being carried out by the pre-empted low priority process.

In the event of a transputer halting because of *HaltOnError*, the links will finish outstanding transfers before shutting down. If **Analyse** is asserted then all inputs continue but outputs will not make another access to memory for data. Memory refresh will continue to take place.

After halting due to the *Error* flag changing from 0 to 1 whilst *HaltOnError* is set, register **I** points two bytes past the instruction which set *Error*. After halting due to the **Analyse** pin being taken high, register **I** points one byte past the instruction being executed. In both cases **I** will be copied to register **A**.

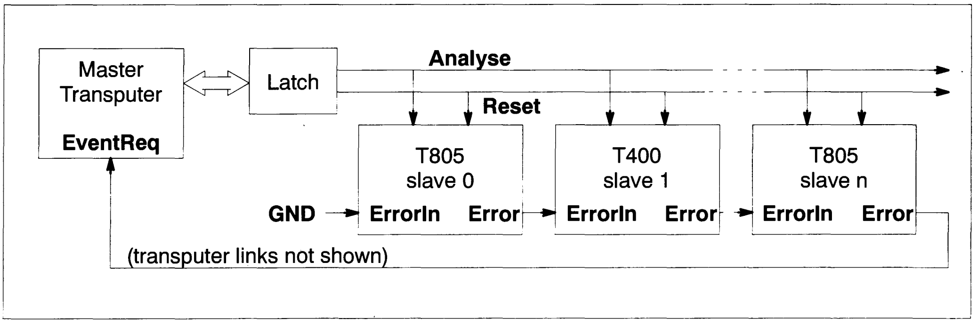


Figure 3.6 Error handling in a multi-transputer system

4 Memory

The IMS T400 has 2 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (page 5.1.12). The transputer can also access 4 Gbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

IMS T400 memory is byte addressed, with words aligned on four-byte boundaries. The least significant byte of a word is the lowest addressed byte.

The bits in a byte are numbered 0 to 7, with bit 0 the least significant. The bytes are numbered from 0, with byte 0 the least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

Internal memory starts at the most negative address #80000000 and extends to #800007FF. User memory begins at #80000070; this location is given the name *MemStart*. An instruction *ldmemstartval* is provided to obtain the value of **MemStart**.

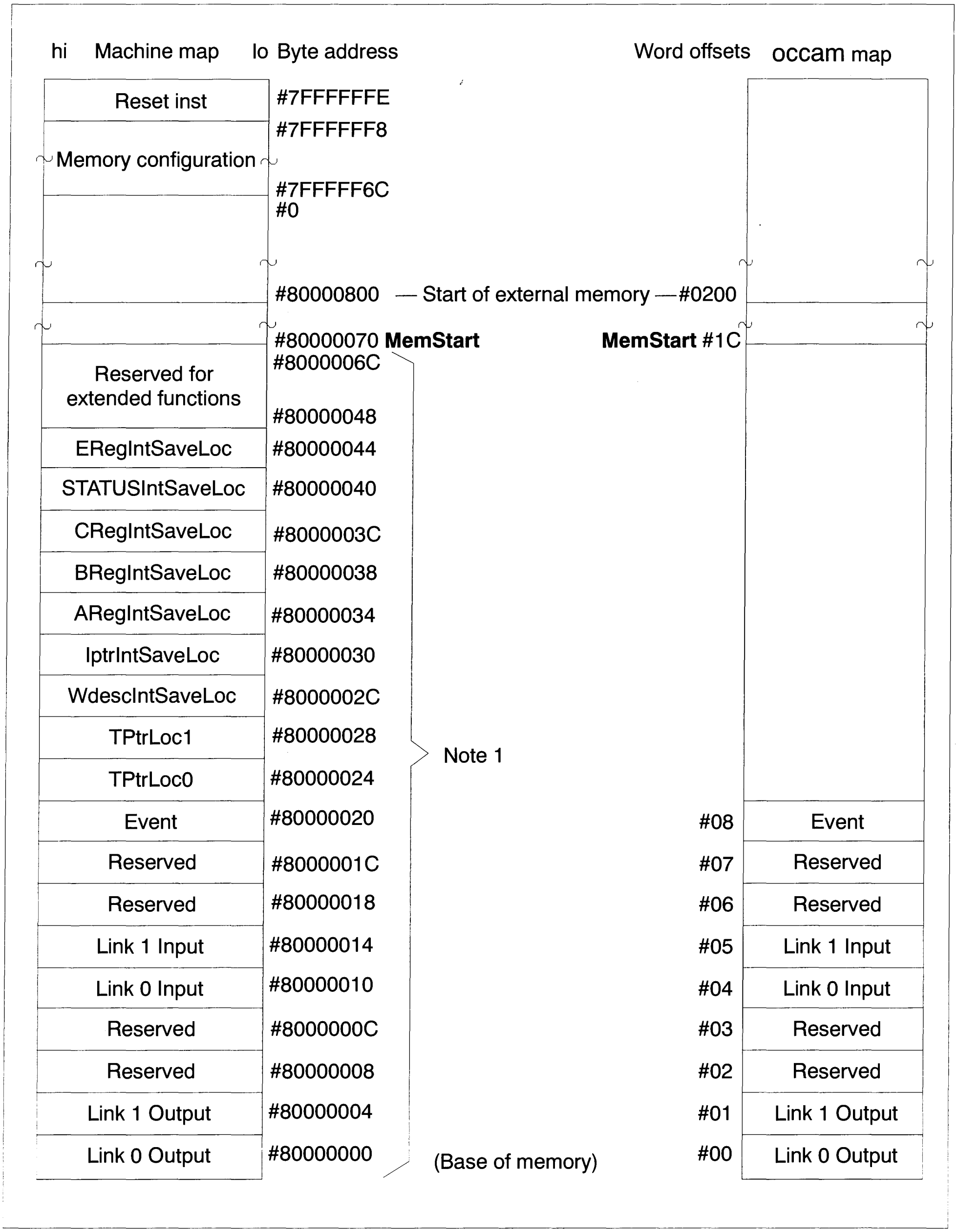
The context of a process in the transputer model involves a workspace descriptor (**WPtr**) and an instruction pointer (**IPtr**). **WPtr** is a word address pointer to a workspace in memory. **IPtr** points to the next instruction to be executed for the process which is the currently executing process. The context switch performed by the breakpoint instruction swaps the **WPtr** and **IPtr** of the currently executing process with the **WPtr** and **IPtr** held above **MemStart**. Two contexts are held above **MemStart**, one for high priority and one for low priority; this allows processes at both levels to have breakpoints. Note that on bootstrapping from a link, these contexts are overwritten by the loaded code. If this is not acceptable, the values should be peeked from memory before bootstrapping from a link.

The reserved area of internal memory below **MemStart** is used to implement link and event channels.

Two words of memory are reserved for timer use, *TPtrLoc0* for high priority processes and *TPtrLoc1* for low priority processes. They either indicate the relevant priority timer is not in use or point to the first process on the timer queue at that priority level.

Values of certain processor registers for the current low priority process are saved in the reserved *IntSaveLoc* locations when a high priority process pre-empts a low priority one. Other locations are reserved for extended features such as block moves.

External memory space starts at #80000800 and extends up through #00000000 to #7FFFFFFF. Memory configuration data and ROM bootstrapping code must be in the most positive address space, starting at #7FFFFF6C and #7FFFFFFE respectively. Address space immediately below this is conventionally used for ROM based code.



hi Machine map lo	Byte address		Word offsets	occam map
Reset inst	#7FFFFFFE			
Memory configuration	#7FFFFFF8 #7FFFFF6C #0			
	#80000800	— Start of external memory —	#0200	
	#80000070 **MemStart**		**MemStart** #1C	
Reserved for extended functions	#8000006C #80000048	Note 1		
ERegIntSaveLoc	#80000044			
STATUSIntSaveLoc	#80000040			
CRegIntSaveLoc	#8000003C			
BRegIntSaveLoc	#80000038			
ARegIntSaveLoc	#80000034			
IptrIntSaveLoc	#80000030			
WdescIntSaveLoc	#8000002C			
TPtrLoc1	#80000028			
TPtrLoc0	#80000024			
Event	#80000020		#08	Event
Reserved	#8000001C		#07	Reserved
Reserved	#80000018		#06	Reserved
Link 1 Input	#80000014		#05	Link 1 Input
Link 0 Input	#80000010		#04	Link 0 Input
Reserved	#8000000C		#03	Reserved
Reserved	#80000008		#02	Reserved
Link 1 Output	#80000004		#01	Link 1 Output
Link 0 Output	#80000000	(Base of memory)	#00	Link 0 Output

Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 278). For details see *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 4.1 IMS T400 memory map

5 External memory interface

The External Memory Interface (EMI) allows access to a 32 bit address space, supporting dynamic and static RAM as well as ROM and EPROM. EMI timing can be configured at **Reset** to cater for most memory types and speeds, and a program is supplied with the Transputer Development System to aid in this configuration.

There are 17 internal configurations which can be selected by a single pin connection (page 300). If none are suitable the user can configure the interface to specific requirements, as shown in page 302.

The external memory cycle is divided into six **Tstates** with the following functions:

T1 Address setup time before address valid strobe.

T2 Address hold time after address valid strobe.

T3 Read cycle tristate or write cycle data setup.

T4 Extendable data setup time.

T5 Read or write data.

T6 Data hold.

Under normal conditions each **Tstate** may be from one to four periods **Tm** long, the duration being set during memory configuration. The default condition on **Reset** is that all **Tstates** are the maximum four periods **Tm** long to allow external initialisation cycles to read slow ROM.

Period **T4** can be extended indefinitely by adding externally generated wait states.

An external memory cycle is always an even number of periods **Tm** in length and the start of **T1** always coincides with a rising edge of **ProcClockOut**. If the total configured quantity of periods **Tm** is an odd number, one extra period **Tm** will be added at the end of **T6** to force the start of the next **T1** to coincide with a rising edge of **ProcClockOut**. This period is designated **E** in configuration diagrams (figure 5.19).

During an internal memory access cycle the external memory interface bus **MemAD2-31** reflects the word address used to access internal RAM, **MemnotWrD0** reflects the read/write operation and **MemnotRfD1** is high; all control strobes are inactive. This is true unless and until a memory refresh cycle or DMA (memory request) activity takes place, when the bus will carry the appropriate external address or data.

The bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 277).

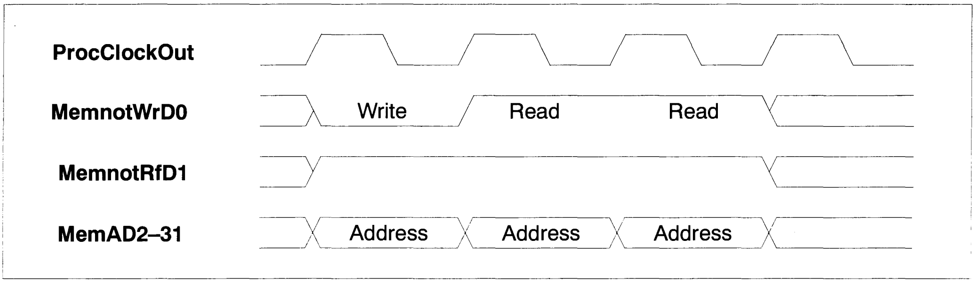


Figure 5.1 IMS T400 bus activity for internal memory cycle

5.1 Pin functions

5.1.1 MemAD2–31

External memory addresses and data are multiplexed on one bus. Only the top 30 bits of address are output on the external memory interface, using pins **MemAD2-31**. They are normally output only during **Tstates T1** and **T2**, and should be latched during this time. The data bus is 32 bits wide. It uses **MemAD2-31** for the top 30 bits and **MemnotRfD1** and **MemnotWrD0** for the lower two bits.

5.1.2 notMemRd

For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Data is read by the transputer on the rising edge of this strobe, and may be removed immediately afterward. If the strobe duration is insufficient it may be extended by adding extra periods **Tm** to either or both of the **Tstates T4** and **T5**. Further extension may be obtained by inserting wait states at the end of **T4**.

5.1.3 MemnotWrD0

During **T1** and **T2** this pin will be low if the cycle is a write cycle, otherwise it will be high. During **Tstates T3** to **T6** it becomes bit 0 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

5.1.4 notMemWrB0–3

Because the transputer uses word addressing, four write strobes are provided; one to write each byte of the word. **notMemWrB0** addresses the least significant byte.

5.1.5 notMemS0–4

To facilitate control of different types of memory and devices, the EMI is provided with five strobe outputs, four of which can be configured by the user. The strobes are conventionally assigned the functions shown in the read and write cycle diagrams, although there is no compulsion to retain these designations.

5.1.6 MemWait

Wait states can be selected by taking **MemWait** high. Externally generated wait states can be added to extend the duration of **T4** indefinitely.

5.1.7 MemnotRfD1

During **T1** and **T2**, this pin is low if the address on **MemAD2-31** is a refresh address, otherwise it is high. During **Tstates T3** to **T6** it becomes bit 1 of the data bus. In both cases it follows the general timing of **MemAD2-31**.

5.1.8 notMemRf

The IMS T400 can be operated with memory refresh enabled or disabled. The selection is made during memory configuration, when the refresh interval is also determined.

5.1.9 RefreshPending

When high, this pin signals that a refresh cycle is pending.

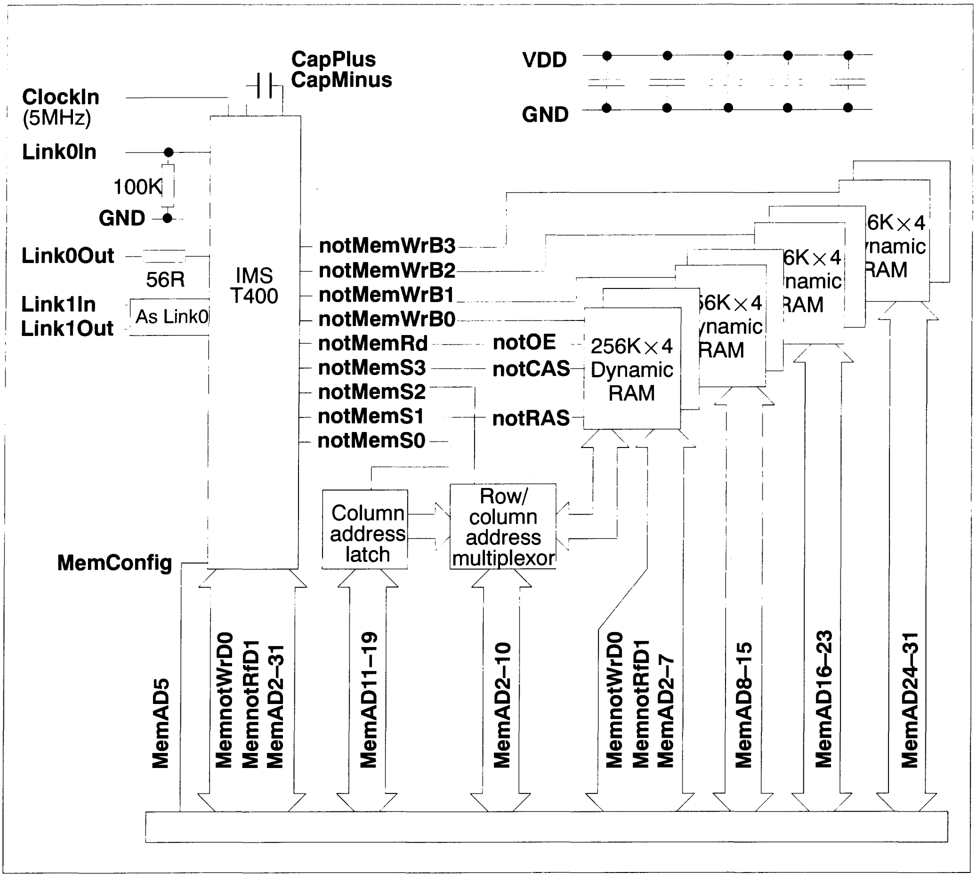


Figure 5.2 IMS T400 dynamic RAM application

5.1.10 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by driving the asynchronous **MemReq** input high. **MemGranted** follows the timing of the bus being tristated and can be used to signal to the device requesting the DMA that it has control of the bus. Note that **MemGranted** changes on the falling edge of **ProcClockOut** and can therefore be sampled to establish control of the bus on the rising edge of **ProcClockOut**.

5.1.11 MemConfig

MemConfig is an input pin used to read configuration data when setting external memory interface (EMI) characteristics.

5.1.12 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn**. Its period is equal to one internal microcode cycle time, and can be derived from the formula

TPCLPCL = **TDCLDCL** / **PLLx**

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (Ordering section).

The time value **Tm** is used to define the duration of **Tstates** and, hence, the length of external memory cycles; its value is exactly half the period of one **ProcClockOut** cycle (0.5***TPCLPCL**), regardless of mark/space ratio of **ProcClockOut**.

Edges of the various external memory strobes coincide with rising or falling edges of **ProcClockOut**. It should be noted, however, that there is a skew associated with each coincidence. The value of skew depends on whether coincidence occurs when the **ProcClockOut** edge and strobe edge are both rising, when both are falling or if either is rising when the other is falling. Timing values given in the strobe tables show the best and worst cases. If a more accurate timing relationship is required, the exact **Tstate** timing and strobe edge to **ProcClockOut** relationships should be calculated and the correct skew factors applied from the edge skew timing table 5.4.

Symbol	Parameter	T400-20		Units	Notes
		Min	Max		
TPCLPCL	**ProcClockOut** period	48	52	ns	2
TPCHPCL	**ProcClockOut** pulse width high	13.5	28.5	ns	2
TPCLPCH	**ProcClockOut** pulse width low	**a**		ns	2, 3, 4
Tm	**ProcClockOut** half cycle	24	26	ns	2
TPCstab	**ProcClockOut** stability		8	%	1,2

Notes

1 Stability is the variation of cycle periods between two consecutive cycles, measured at corresponding points on the cycles.

2 This parameter is sampled and not 100% tested.

3 **a** is TPCLPCL – TPCHPCL.

4 This is a nominal value.

Table 5.1 **ProcClockOut**

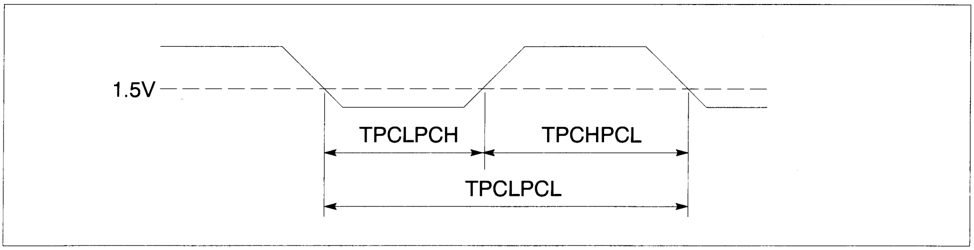


Figure 5.3 IMS T400 **ProcClockOut** timing

5.2 Read cycle

Byte addressing is carried out internally by the transputer for read cycles. For a read cycle the read strobe **notMemRd** is low during **T4** and **T5**. Read cycle data may be set up on the data bus at any time after the start of **T3**, but must be valid when the transputer reads it at the end of **T5**. Data may be removed any time during **T6**, but must be off the bus no later than the end of that period.

notMemS0 is a fixed format strobe. Its leading edge is always coincident with the start of **T2** and its trailing edge always coincident with the end of **T5**.

The leading edge of **notMemS1** is always coincident with the start of **T2**, but its duration may be configured to be from zero to 31 periods **Tm**. Regardless of the configured duration, the strobe will terminate no later than the end of **T6**. The strobe is sometimes programmed to extend beyond the normal end of **Tmx**. When wait states are inserted into an EMI cycle the end of **Tmx** is delayed, but the potential active duration of the strobe is not altered. Thus the strobe can be configured to terminate relatively early under certain conditions (page 293). If **notMemS1** is configured to be zero it will never go low.

notMemS2, **notMemS3** and **notMemS4** are identical in operation. They all terminate at the end of **T5**, but the start of each can be delayed from one to 31 periods **Tm** beyond the start of **T2**. If the duration of one of these strobes would take it past the end of **T5** it will stay high. This can be used to cause a strobe to become active only when wait states are inserted. If one of these strobes is configured to zero it will remain low during **T1**–**T5** and only go high during the first **Tm** of **T6**. Figure 5.6 shows the effect of **Wait** on strobes in more detail; each division on the scale is one period **Tm**.

In the read cycle timing diagrams **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**.

Symbol	Parameter	T400-20		Units	Notes
		Min	Max		
TaZdV	Address tristate to data valid	0		ns	3
TdVRdH	Data setup before read	25		ns	
TRdHdX	Data hold after read	0		ns	3
TS0LRdL	**notMemS0** before start of read	**a**–4	**a**+4	ns	1
TS0HRdH	End of read from end of **notMemS0**	–4	4	ns	
TRdLRdH	Read period	**b**–3	**b**+5	ns	2

Notes

1 **a** is total of **T2**+**T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

2 **b** is total of **T4**+**Twait**+**T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

3 Guaranteed, but not tested.

Table 5.2 Read

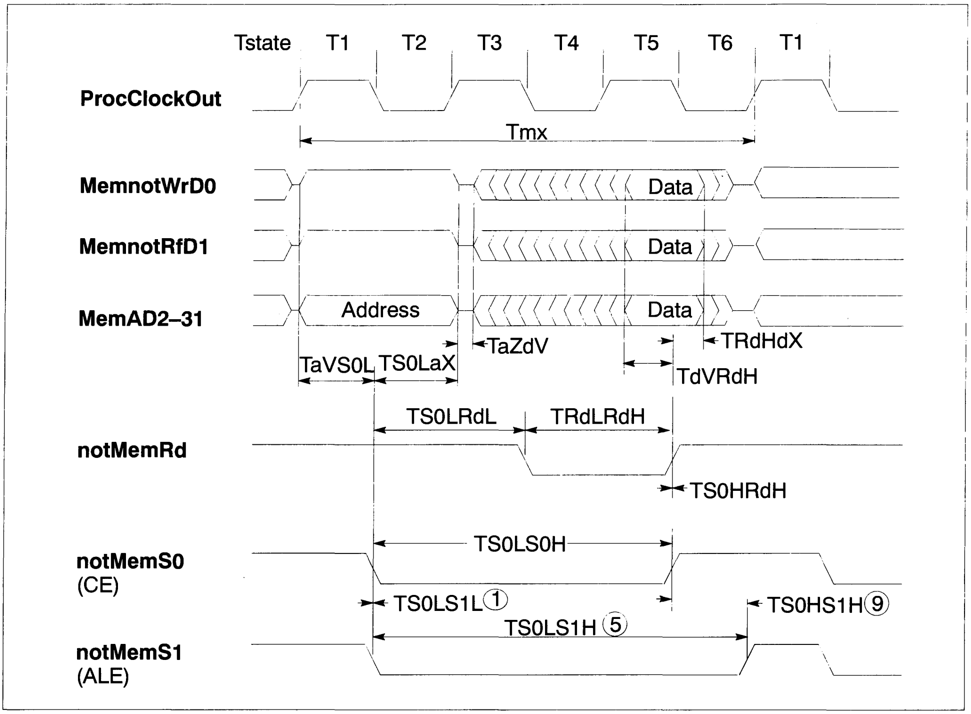


Figure 5.4 IMS T400 external read cycle: static memory

Symbol	n	Parameter	T400-20		Units	Notes
			Min	Max		
TaVS0L		Address setup before **notMemS0**	**a**–8		ns	1
TS0LaX		Address hold after **notMemS0**	**b**–8	**b**+8	ns	2
TS0LS0H		**notMemS0** pulse width low	**c**–5	**c**+6	ns	3
TS0LS1L	1	**notMemS1** from **notMemS0**	–4	4	ns	10
TS0LS1H	5	**notMemS1** end from **notMemS0**	**d**–1	**d**+9	ns	4,6
TS0HS1H	9	**notMemS1** end from **notMemS0** end	**e**–8	**e**+4	ns	5,6
TS0LS2L	2	**notMemS2** delayed after **notMemS0**	**f**–6	**f**+5	ns	7
TS0LS2H	6	**notMemS2** end from **notMemS0**	**c**–5	**c**+7	ns	3,10
TS0HS2H	10	**notMemS2** end from **notMemS0** end	–4	7	ns	10
TS0LS3L	3	**notMemS3** delayed after **notMemS0**	**f**–6	**f**+5	ns	7
TS0LS3H	7	**notMemS3** end from **notMemS0**	**c**–5	**c**+7	ns	3,10
TS0HS3H	11	**notMemS3** end from **notMemS0** end	–4	7	ns	10
TS0LS4L	4	**notMemS4** delayed after **notMemS0**	**f**–6	**f**+5	ns	7
TS0LS4H	8	**notMemS4** end from **notMemS0**	**c**–5	**c**+7	ns	3,10
TS0HS4H	12	**notMemS4** end from **notMemS0** end	–4	7	ns	10
Tmx		Complete external memory cycle				8,9

Notes

1 **a** is **T1** where **T1** can be from one to four periods **Tm** in length.

2 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

3 **c** is total of **T2**+**T3**+**T4**+**Twait**+**T5** where **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be any number of periods **Tm** in length.

4 **d** can be from zero to 31 periods **Tm** in length.

5 **e** can be from –27 to +4 periods **Tm** in length.

6 If the configuration would cause the strobe to remain active past the end of **T6** it will go high at the end of **T6**. If the strobe is configured to zero periods **Tm** it will remain high throughout the complete cycle **Tmx**.

7 **f** can be from zero to 31 periods **Tm** in length. If this length would cause the strobe to remain active past the end of **T5** it will go high at the end of **T5**. If the strobe value is zero periods **Tm** it will remain low throughout **T1** to **T5** and only go high during the first **Tm** of **T6**.

8 **Tmx** is one complete external memory cycle comprising the total of **T1**+**T2**+**T3**+**T4**+**Twait**+**T5**+**T6** where **T1**, **T2**, **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length, **T6** can be from one to five periods **Tm** in length and **Twait** may be zero or any number of periods **Tm** in length.

9 Guaranteed, but not tested.

10 Sampled, not 100% tested.

Table 5.3 IMS T400 strobe timing

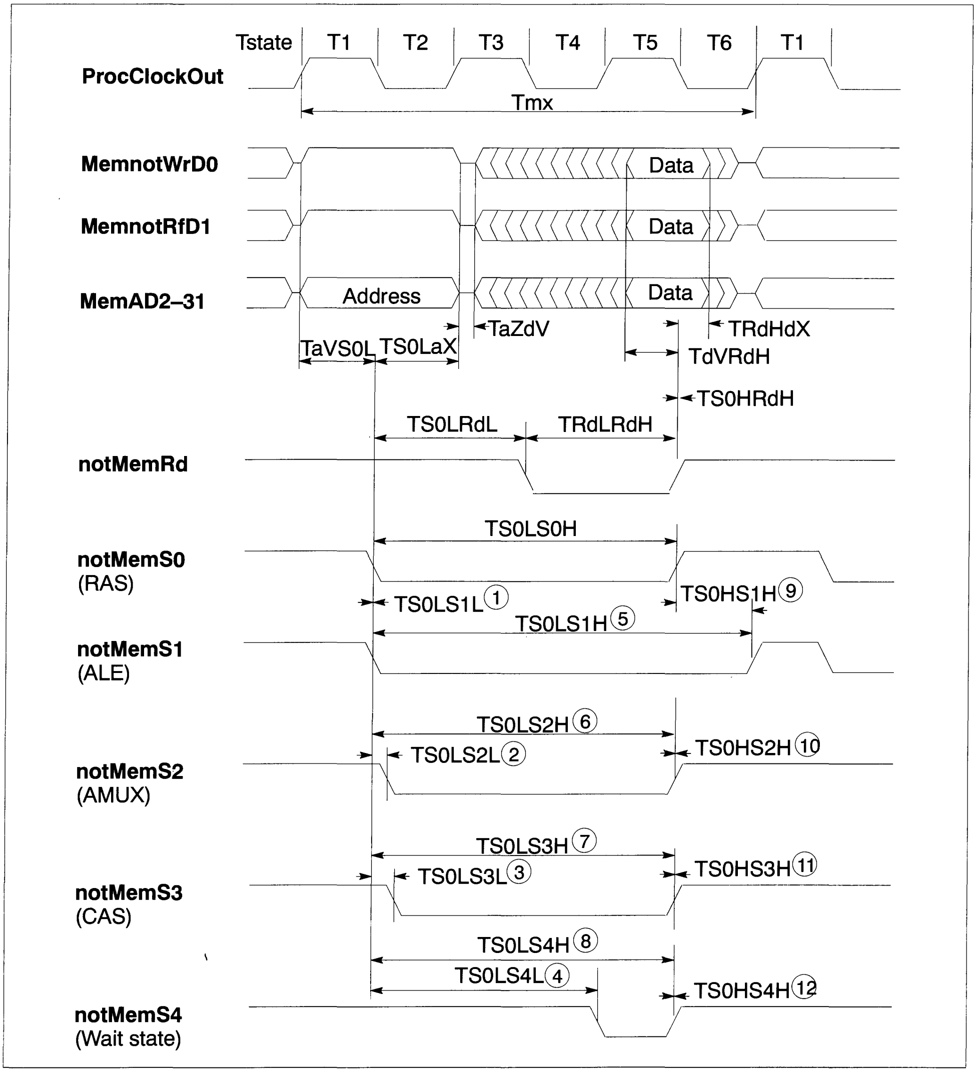


Figure 5.5 IMS T400 external read cycle: dynamic memory

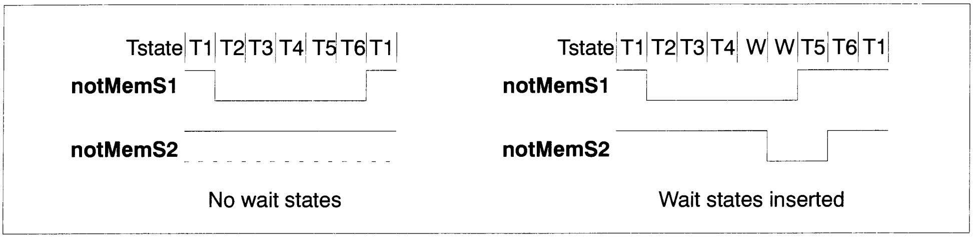


Figure 5.6 IMS T400 effect of wait states on strobes

Symbol	Parameter	T400-20		Units	Notes
		Min	Max		
TPCHS0H	**notMemS0** rising from **ProcClockOut** rising	–6	4	ns	1
TPCLS0H	**notMemS0** rising from **ProcClockOut** falling	–5	10	ns	1
TPCHS0L	**notMemS0** falling from **ProcClockOut** rising	–8	3	ns	1
TPCLS0L	**notMemS0** falling from **ProcClockOut** falling	–5	7	ns	1

Notes

1 Sampled, not 100% tested.

Table 5.4 Strobe S0 to **ProcClockOut** skew

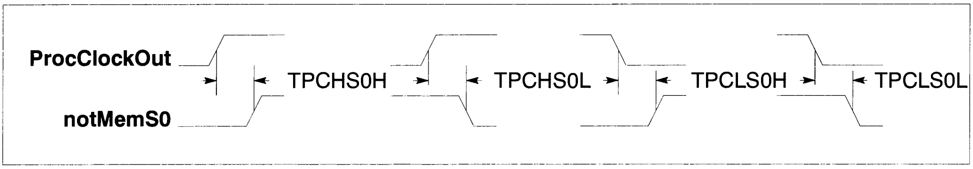


Figure 5.7 IMS T400 skew of **notMemS0** to **ProcClockOut**

5.3 Write cycle

For write cycles the relevant bytes in memory are addressed by the write strobes **notMemWrB0-3**. If a particular byte is not to be written, then the corresponding data outputs are tristated.

For a write cycle pin **MemnotWrD0** will be low during **T1** and **T2**. Write data is placed on the bus at the start of **T3** and removed at the end of **T6**. If **T6** is extended to force the next cycle **Tmx** (page 284) to start on a rising edge of **ProcClockOut**, data will be valid during this time also.

The transputer has both early and late write cycle modes. For a late write cycle the relevant write strobes **notMemWrB0-3** are low during **T4** and **T5**; for an early write they are also low during **T3**. Data should be latched into memory on the rising edge of the strobes in both cases, although it is valid until the end of **T6**. If the strobe duration is insufficient, it may be extended at configuration time by adding extra periods **Tm** to either or both of **Tstates T4** and **T5** for both early and late modes. For an early cycle they may also be added to **T3**. Further extension may be obtained by inserting wait states at the end of **T4**. If the data hold time is insufficient, extra periods **Tm** may be added to **T6** to extend it.

In the write cycle timing diagram **ProcClockOut** is included as a guide only; it is shown with each **Tstate** configured to one period **Tm**. The strobe is inactive during internal memory cycles.

Symbol	Parameter	T400-20		Units	Notes
		Min	Max		
TdVWrH	Data setup before write	**d**–7	**d**+10	ns	1,5
TWrHdX	Data hold after write	**a**–10	**a**+5	ns	1,2
TS0LWrL	**notMemS0** before start of early write	**b**–5	**b**+5	ns	1,3
	notMemS0 before start of late write	**c**–5	**c**+5	ns	1,4
TS0HWrH	End of write from end of **notMemS0**	–5	4	ns	1,7
TWrLWrH	Early write pulse width	**d**–4	**d**+7	ns	1,5
	Late write pulse width	**e**–4	**e**+7	ns	1,6

Notes

1 Timing is for all write strobes **notMemWrB0-3**.

2 **a** is **T6** where **T6** can be from one to five periods **Tm** in length.

3 **b** is **T2** where **T2** can be from one to four periods **Tm** in length.

4 **c** is total of **T2+T3** where **T2**, **T3** can be from one to four periods **Tm** each in length.

5 **d** is total of **T3+T4+Twait+T5** where **T3**, **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

6 **e** is total of **T4+Twait+T5** where **T4**, **T5** can be from one to four periods **Tm** each in length and **Twait** may be zero or any number of periods **Tm** in length.

7 Sampled, not 100% tested.

Table 5.5 Write

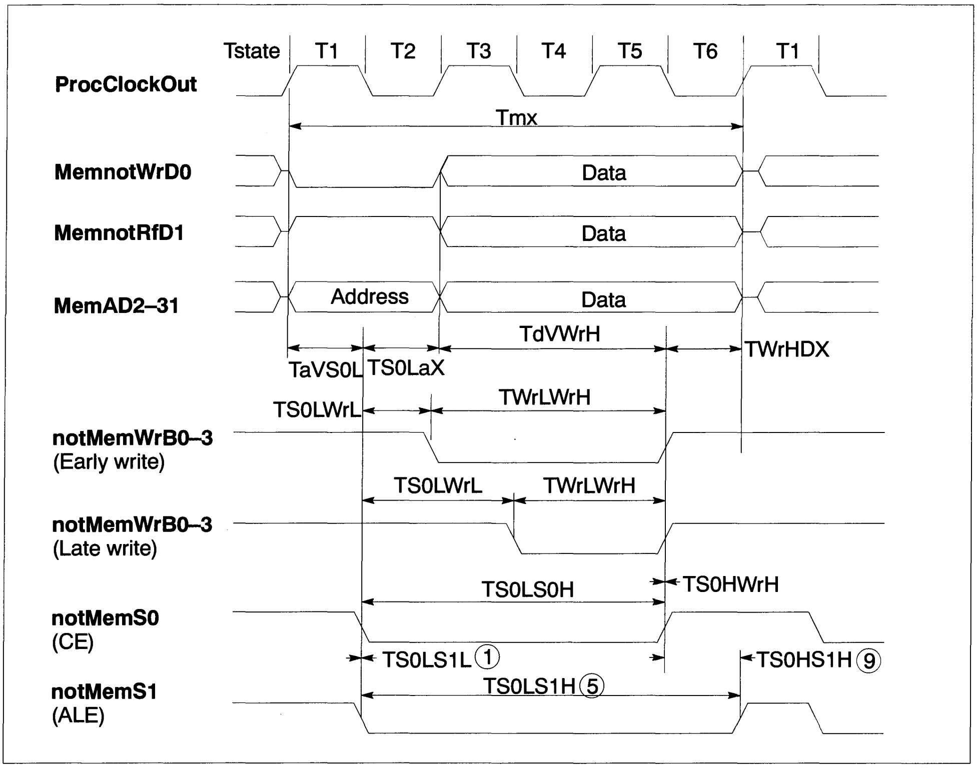


Figure 5.8 IMS T400 external write cycle

5.4 Wait

Taking **MemWait** high with the timing shown (figure 5.9) will extend the duration of **T4**. **MemWait** is sampled close to the falling edge of **ProcClockOut** during a **T3** or **T4** period, prior to, but not at the end of **T4**. By convention, **notMemS4** is used to synchronize wait state insertion. If this or another strobe is used, its delay should be such as to take the strobe low an even number of periods **Tm** after the start of **T1**, to coincide with a rising edge of **ProcClockOut**.

MemWait may be kept high indefinitely, although if dynamic memory refresh is used it should not be kept high long enough to interfere with refresh timing. **MemWait** operates normally during all cycles, including refresh and configuration cycles. It does not affect internal memory access in any way.

If the start of **T5** would coincide with a falling edge of **ProcClockOut** an extra wait period **Tm** (**EW**) is generated by the EMI to force coincidence with a rising edge. Rising edge coincidence is only forced if wait states are added, otherwise coincidence with a falling edge is permitted.

		IMS T400-20			
Symbol	Parameter	Min	Max	Units	Notes
TPCLWtH	Wait setup	10		ns	1,2
TPCLWtL	Wait hold	8		ns	1,2
TWtLWtH	Delay before re-assertion of Wait	50		ns	3

Notes

1 **ProcClockOut** load should not exceed 50pf.

2 If wait period exceeds refresh interval, refresh cycles will be lost.

3 Guaranteed, but not tested.

Table 5.6 IMS T400 memory wait

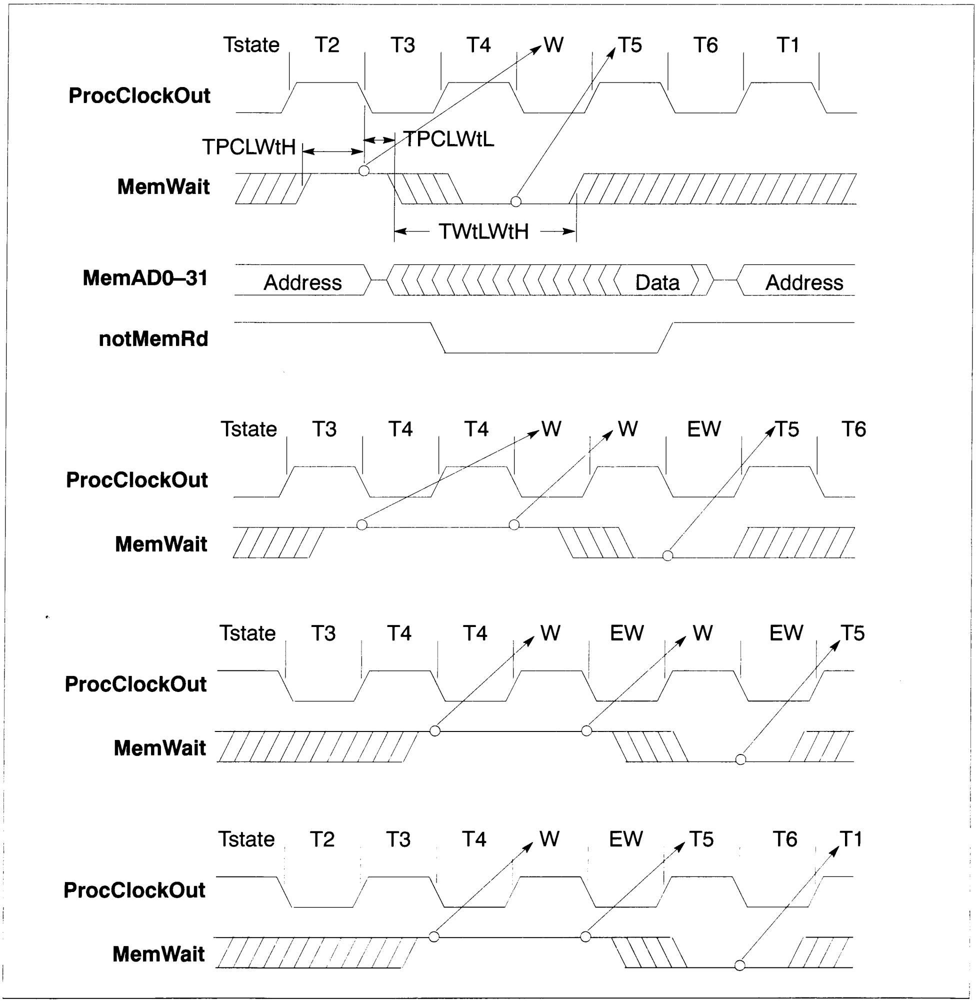


Figure 5.9 IMS T400 memory wait timing

5.5 Memory refresh

The **RefreshPending** pin is asserted high when the external memory interface is about to perform a refresh cycle. It remains high until the refresh cycle is started by the transputer. The mimimum time for the **RefreshPending** pin to be high is for one cycle of **ProcClockOut** (two periods **Tm**), when the EMI was not about to perform a memory read or write. If the EMI was held in the tristate condition with **MemGranted** asserted, then **RefreshPending** will be asserted when the refresh controller in the EMI is ready to perform a refresh. **MemReq** may be re-asserted any time after the commencement of the refresh cycle. **RefreshPending** changes state near the rising edge of **ProcClockOut** and can therefore be sampled by the falling edge of **ProcClockOut**.

If no DMA is active then refresh will be performed following the end of the current internal or external memory cycle. If DMA is active the transputer will wait for DMA to terminate before commencing the refresh cycle. Unlike **MemnotRfD1**, **RefreshPending** is never tristated and can thus be interrogated by the DMA device; the DMA cycle can then be suspended, at the discretion of the DMA device, to allow refresh to take place.

The simple circuit of Figure 5.10 will suspend DMA requests from the external logic when **RefreshPending** is asserted, so that a memory refresh cycle can be performed. DMA is restored on completion of the refresh cycle. The transputer will not perform an external memory cycle other than a refresh cycle, using this method, until the requesting device removes its DMA request.

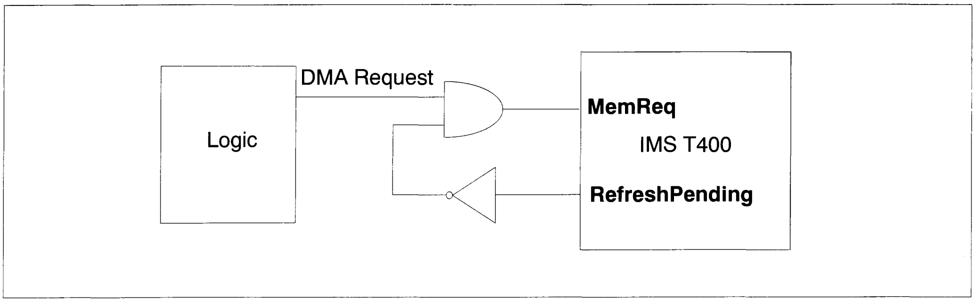


Figure 5.10 IMS T400 refresh with DMA

When refresh is disabled no refresh cycles occur. During the post-**Reset** period eight dummy refresh cycles will occur with the appropriate timing but with no bus or strobe activity.

A refresh cycle uses the same basic external memory timing as a normal external memory cycle, except that it starts two periods **Tm** before the start of **T1**. If a refresh cycle is due during an external memory access, it will be delayed until the end of that external cycle. Two extra periods **Tm** (periods **R** in the diagram) will then be inserted between the end of **T6** of the external memory cycle and the start of **T1** of the refresh cycle itself. The refresh address and various external strobes become active approximately one period **Tm** before **T1**. Bus signals are active until the end of **T2**, whilst **notMemRf** remains active until the end of **T6**.

For a refresh cycle, **MemnotRfD1** goes low when **notMemRf** goes low and **MemnotWrD0** goes high with the same timing as **MemnotRfD1**. All the address lines share the same timing, but only **MemAD2-11** give the refresh address. **MemAD12-30** stay high during the address period, whilst **MemAD31** remains low. Refresh cycles generate strobes **notMemS0-4** with timing as for a normal external cycle, but **notMemRd** and **notMemWrB0-3** remain high. **MemWait** operates normally during refresh cycles.

Refresh cycles do not interrupt internal memory accesses, although the internal addresses cannot be reflected on the external bus during refresh.

Symbol	Parameter	T400-20		Units	Notes
		Min	Max		
TRfLRfH	Refresh pulse width low	**a**–2	**a**+9	ns	1
TRaVS0L	Refresh address setup before **notMemS0**	**b**–12		ns	
TRfLS0L	Refresh indicator setup before **notMemS0**	**b**–4	**b**+6	ns	2

Notes

1 **a** is total **Tmx**+**Tm**.

2 **b** is total **T1**+**Tm** where **T1** can be from one to four periods **Tm** in length.

Table 5.7 Memory refresh

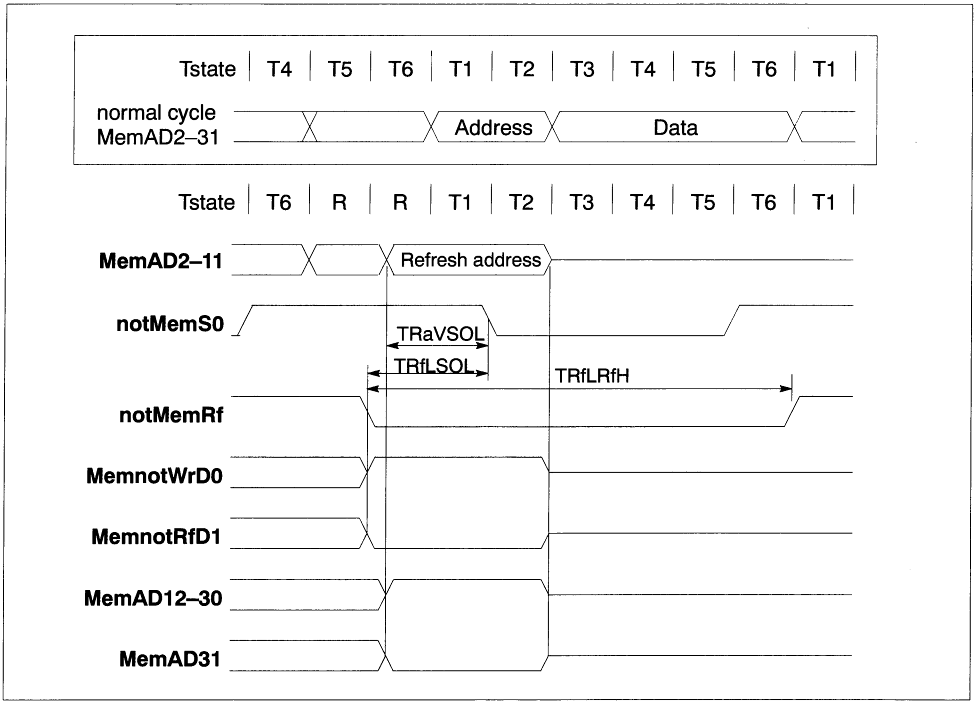


Figure 5.11 IMS T400 refresh cycle timing

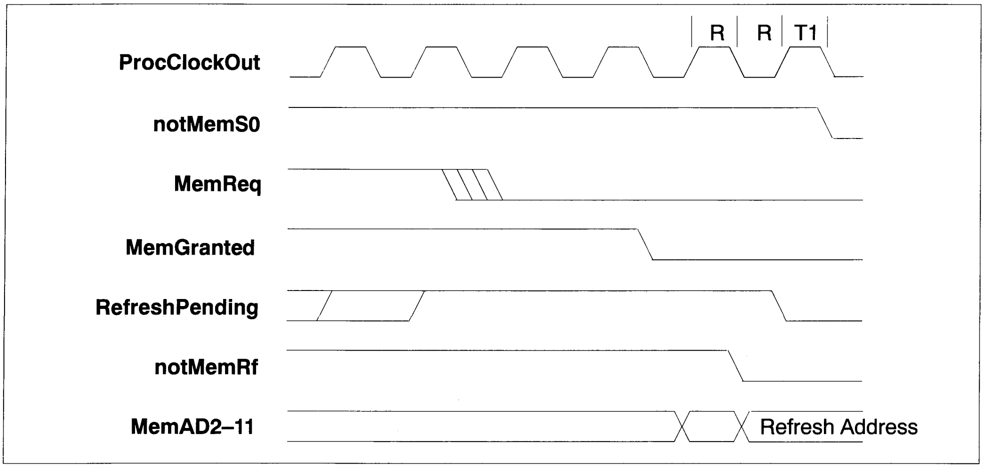


Figure 5.12 IMS T400 Refresh Pending timing

5.6 Direct memory access

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. The transputer samples **MemReq** just before falling edges of **ProcClockOut**. To guarantee taking over the bus immediately following either a refresh or external memory cycle, **MemReq** must be sampled at least four periods **Tm** before the end of **T6**. In the absence of an external memory cycle, the address bus is tristated two periods **Tm** after the **ProcClockOut** rising edge which follows the sample.

Removal of **MemReq** is sampled just before falling edges of **ProcClockOut** and **MemGranted** is removed synchronously with the second falling edge of **ProcClockOut** which follows the sample. If accurate timing of DMA is required, the setup time relative to **ProcClockOut** must be met. Further external bus activity, either refresh, external cycles or reflection of internal cycles, will commence at the next but one rising edge of **ProcClockOut**.

The strobes (**notMemS0–4** and **notMemWrB0–3**) are left in their inactive states during DMA. DMA cannot interrupt a refresh or external memory cycle, and outstanding refresh cycles will occur before the bus is released to DMA. DMA does not interfere with internal memory cycles in any way, although a program running in internal memory would have to wait for the end of DMA before accessing external memory. DMA cannot access internal memory. If DMA extends longer than one refresh interval (Memory Refresh Configuration Coding, table 5.11), the DMA user becomes responsible for refresh (see section 5.5). DMA may also inhibit an internally running program from accessing external memory.

DMA allows a bootstrap program to be loaded into external RAM ready for execution after reset. If **MemReq** is held high throughout reset, **MemGranted** will be asserted before the bootstrap sequence begins. **MemReq** must be high at least one period **TDCLDCL** of **ClockIn** before **Reset**. The circuit should be designed to ensure correct operation if **Reset** could interrupt a normal DMA cycle.

Symbol	Parameter	T400-20		Units	Note
		Min	Max		
TMRHPCL	**MemReq** setup before **ProcClockOut** falling	3	14	ns	1
TPCLMGH	**MemReq** response time	96	110	ns	2
TMRLPCL	**Memreq** removal before **ProcClockOut** falling	4	16		
TPCLMGL	**MemReq** end response time	50	66	ns	
TADZMGH	Bus tristate before **MemGranted**	0	27	ns	
TMGLADV	Bus active after end of **MemGranted**	0	32	ns	

Notes

1 Setup time need only be met to guarantee sampling on this edge.

2 If an external cycle is active, maximum time could be (1 EMI cycle **Tmx**)+(1 refresh cycle **TRfLRfH**)+(6 periods **Tm**).

Table 5.8 Memory request

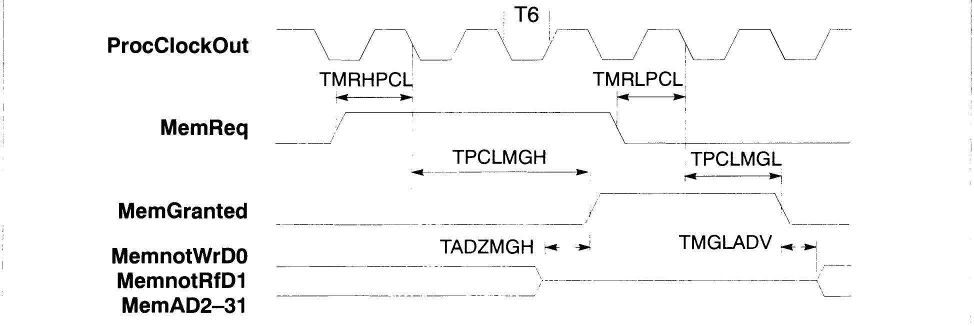
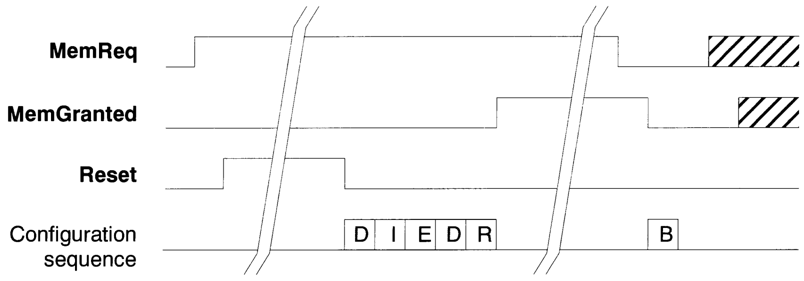


Figure 5.13 IMS T400 memory request timing



- **D** Pre– and post–configuration delays (figure 3.3)
- **I** Internal configuration sequence
- **E** External configuration sequence
- **R** Initial refresh sequence
- **B** Bootstrap sequence

Figure 5.14 IMS T400 DMA sequence at reset

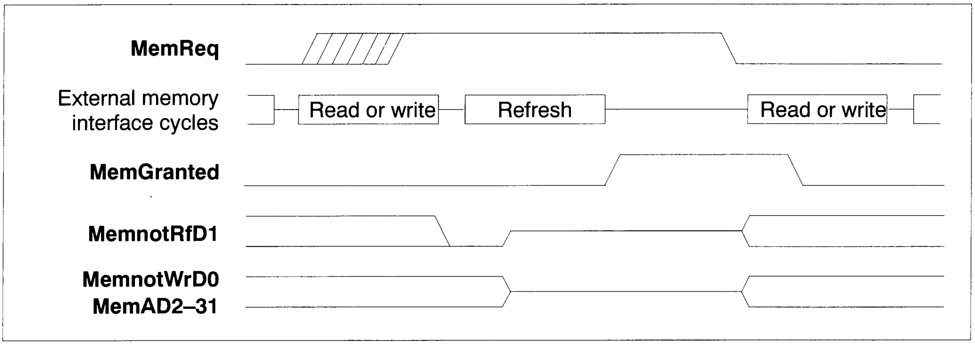


Figure 5.15 IMS T400 operation of **MemReq**, **MemGranted** with external, refresh memory cycles

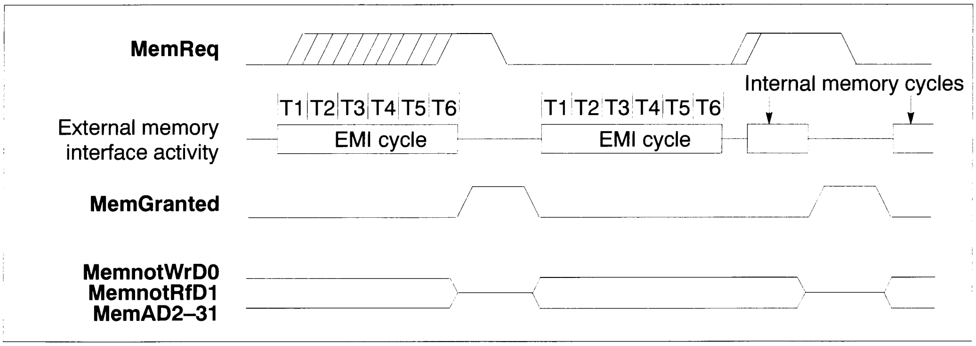


Figure 5.16 IMS T400 operation of **MemReq**, **MemGranted** with external, internal memory cycles

5.7 Memory configuration

MemConfig is an input pin used to read configuration data when setting external memory interface (EMI) characteristics. It is read by the processor on two occasions after **Reset** goes low; first to check if one of the preset internal configurations is required, then to determine a possible external configuration.

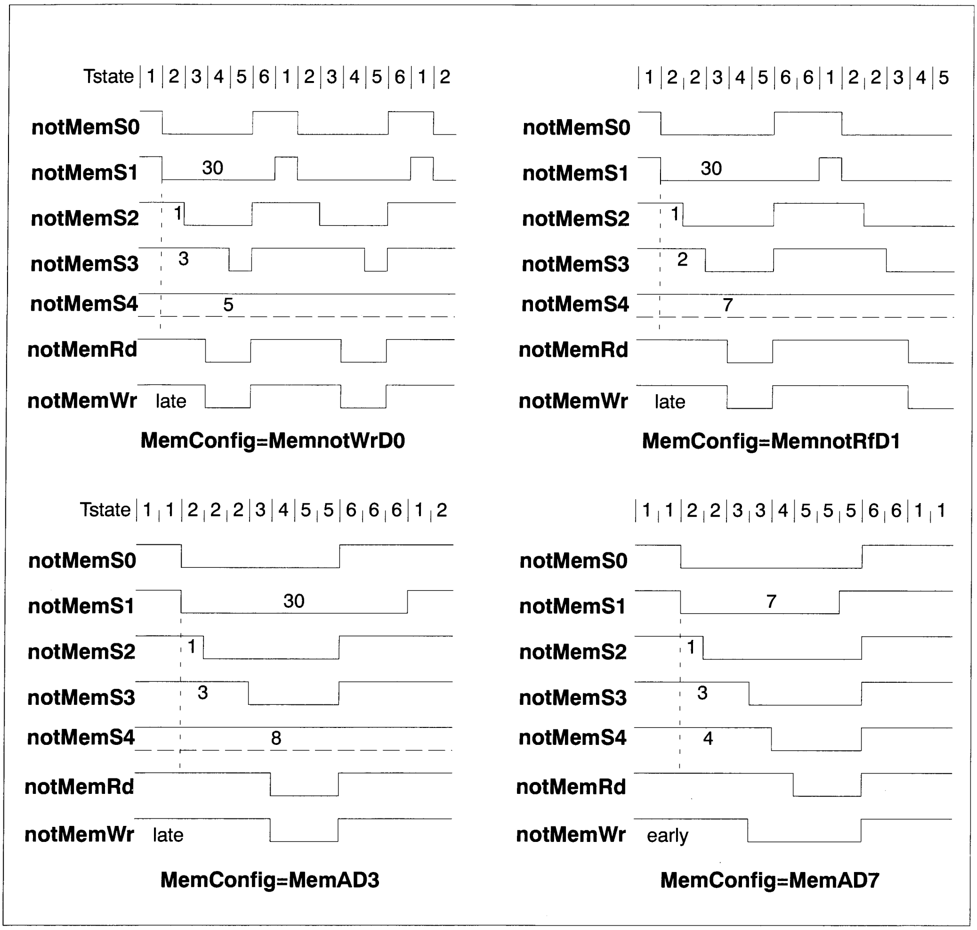
5.7.1 Internal configuration

The internal configuration scan comprises 64 periods **TDCLDCL** of **ClockIn** during the internal scan period of 144 **ClockIn** periods. **MemnotWrD0**, **MemnotRfD1** and **MemAD2-32** are all high at the beginning of the scan. Starting with **MemnotWrD0**, each of these lines goes low successively at intervals of two **ClockIn** periods and stays low until the end of the scan. If one of these lines is connected to **MemConfig** the preset internal configuration mode associated with that line will be used as the EMI configuration. The default configuration is that defined in the table for **MemAD31**; connecting **MemConfig** to **VDD** will also produce this default configuration. Note that only 17 of the possible configurations are valid, all others remain at the default configuration.

Pin	Duration of each Tstate periods Tm						Strobe coefficient				Write cycle	Refresh interval	Cycle time
	T1	T2	T3	T4	T5	T6	s1	s2	s3	s4	type	ClockIn cycles	Proc cycles
MemnotWrD0	1	1	1	1	1	1	30	1	3	5	late	72	3
MemnotRfD1	1	2	1	1	1	2	30	1	2	7	late	72	4
MemAD2	1	2	1	1	2	3	30	1	2	7	late	72	5
MemAD3	2	3	1	1	2	3	30	1	3	8	late	72	6
MemAD4	1	1	1	1	1	1	3	1	2	3	early	72	3
MemAD5	1	1	2	1	2	1	5	1	2	3	early	72	4
MemAD6	2	1	2	1	3	1	6	1	2	3	early	72	5
MemAD7	2	2	2	1	3	2	7	1	3	4	early	72	6
MemAD8	1	1	1	1	1	1	30	1	2	3	early	†	3
MemAD9	1	1	2	1	2	1	30	2	5	9	early	†	4
MemAD10	2	2	2	2	4	2	30	2	3	8	late	72	7
MemAD11	3	3	3	3	3	3	30	2	4	13	late	72	9
MemAD12	1	1	2	1	2	1	4	1	2	3	early	72	4
MemAD13	2	1	2	1	2	2	5	1	2	3	early	72	5
MemAD14	2	2	2	1	3	2	6	1	3	4	early	72	6
MemAD15	2	1	2	3	3	3	8	1	2	3	early	72	7
MemAD31	4	4	4	4	4	4	31	30	30	18	late	72	12

† Provided for static RAM only.

Table 5.9 IMS T400 internal configuration coding



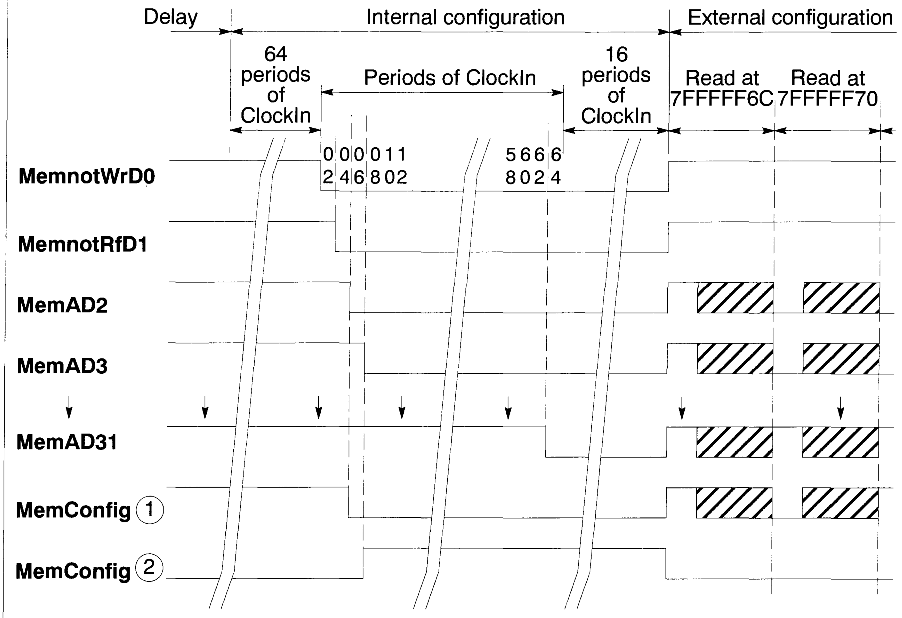
MemConfig=MemnotWrD0

MemConfig=MemnotRfD1

MemConfig=MemAD3

MemConfig=MemAD7

Figure 5.17 IMS T400 internal configuration



① Internal configuration: **MemConfig** connected to **MemAD2**

② External configuration: **MemConfig** connected to inverse of **MemAD3**

Figure 5.18 IMS T400 internal configuration scan

5.7.2 External configuration

If **MemConfig** is held low until **MemnotWrD0** goes low the internal configuration is ignored and an external configuration will be loaded instead. An external configuration scan always follows an internal one, but if an internal configuration occurs any external configuration is ignored.

The external configuration scan comprises 36 successive external read cycles, using the default EMI configuration preset by **MemAD31**. However, instead of data being read on the data bus as for a normal read cycle, only a single bit of data is read on **MemConfig** at each cycle. Addresses put out on the bus for each read cycle are shown in table 5.10, and are designed to address ROM at the top of the memory map. The table shows the data to be held in ROM; data required at the **MemConfig** pin is the inverse of this.

MemConfig is typically connected via an inverter to **MemnotWrD0**. Data bit zero of the least significant byte of each ROM word then provides the configuration data stream. By switching **MemConfig** between various data bus lines up to 32 configurations can be stored in ROM, one per bit of the data bus. **MemConfig** can be permanently connected to a data line or to **GND**. Connecting **MemConfig** to **GND** gives all **Tstates** configured to four periods; **notMemS1** pulse of maximum duration; **notMemS2-4** delayed by maximum; refresh interval 72 periods of **ClockIn**; refresh enabled; late write.

The external memory configuration table 5.10 shows the contribution of each memory address to the 13 configuration fields. The lowest 12 words (#7FFFFF6C to #7FFFFF98, fields 1 to 6) define the number of extra periods **Tm** to be added to each **Tstate**. If field 2 is 3 then three extra periods will be added to **T2** to extend it to the maximum of four periods.

The next five addresses (field 7) define the duration of **notMemS1** and the following fifteen (fields 8 to 10) define the delays before strobes **notMemS2-4** become active. The five bits allocated to each strobe allow durations of from 0 to 31 periods **Tm**, as described in strobes page 284.

Addresses #7FFFFFEC to #7FFFFFF4 (fields 11 and 12) define the refresh interval and whether refresh is to be used, whilst the final address (field 13) supplies a high bit to **MemConfig** if a late write cycle is required.

The columns to the right of the coding table show the values of each configuration bit for the four sample external configuration diagrams. Note the inclusion of period **E** at the end of **T6** in some diagrams. This is inserted to bring the start of the next **Tstate T1** to coincide with a rising edge of **ProcClockOut** (page 285).

Wait states **W** have been added to show the effect of them on strobe timing; they are not part of a configuration. In each case which includes wait states, two wait periods are defined. This shows that if a wait state would cause the start of **T5** to coincide with a falling edge of **ProcClockOut**, another period **Tm** is generated by the EMI to force it to coincide with a rising edge of **ProcClockOut**. This coincidence is only necessary if wait states are added, otherwise coincidence with a falling edge is permitted. Any configuration memory access is only permitted to be extended using wait, up to a total of 14 **ClockIn** periods.

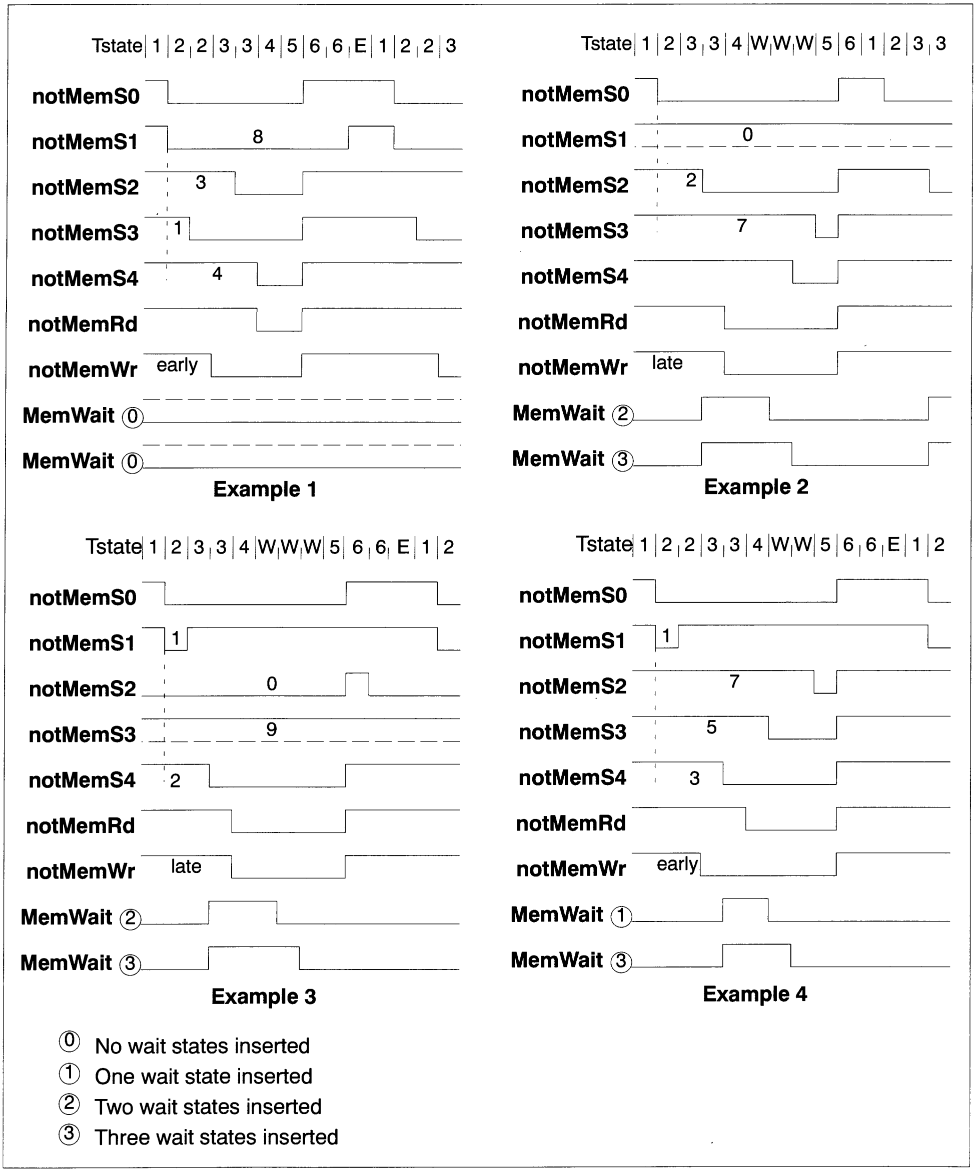
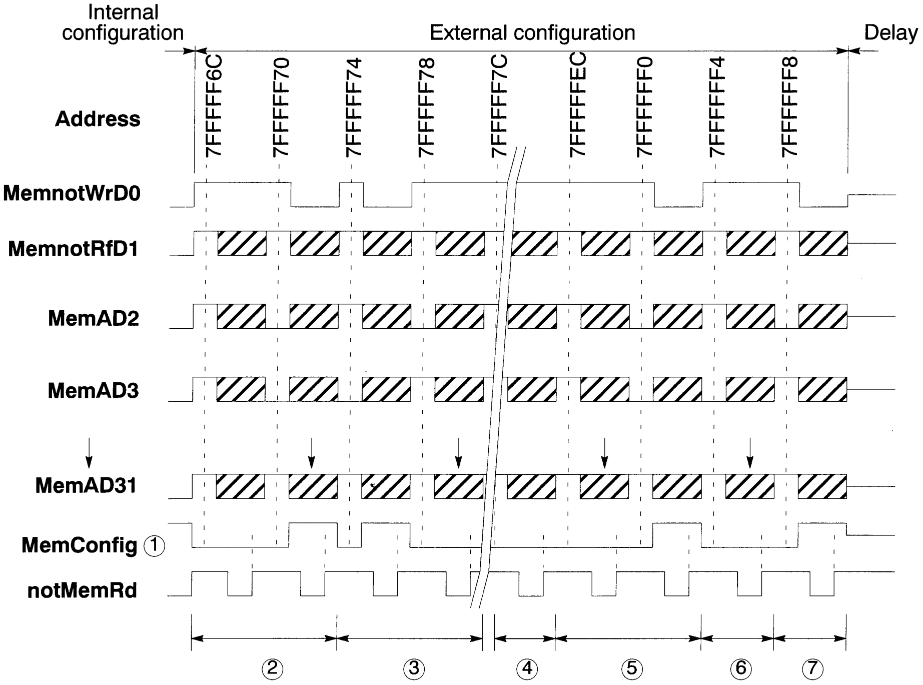


Figure 5.19 IMS T400 external configuration



① **MemConfig** connected to inverse of **MemnotWrD0**

② Configuration field 1; **T1** configured for 2 periods **Tm**

③ Configuration field 2; **T2** configured for 2 periods **Tm**

④ Configuration field 10; most significant bit of **notMemS4** configured high

⑤ Configuration field 11; refresh interval configured for 36 periods **ClockIn**

⑥ Configuration field 12; refresh enabled

⑦ Configuration field 13; early write cycle

Figure 5.20 IMS T400 external configuration scan

Scan cycle	Mem AD address	Field	Function	Example diagram			
				1	2	3	4
1	7FFFFF6C	1	T1 least significant bit	0	0	0	0
2	7FFFFF70	1	T1 most significant bit	0	0	0	0
3	7FFFFF74	2	T2 least significant bit	1	0	0	1
4	7FFFFF78	2	T2 most significant bit	0	0	0	0
5	7FFFFF7C	3	T3 least significant bit	1	1	1	1
6	7FFFFF80	3	T3 most significant bit	0	0	0	0
7	7FFFFF84	4	T4 least significant bit	0	0	0	0
8	7FFFFF88	4	T4 most significant bit	0	0	0	0
9	7FFFFF8C	5	T5 least significant bit	0	0	0	0
10	7FFFFF90	5	T5 most significant bit	0	0	0	0
11	7FFFFF94	6	T6 least significant bit	1	0	1	1
12	7FFFFF98	6	T6 most significant bit	0	0	0	0
13	7FFFFF9C	7	notMemS1 least significant bit	0	0	1	1
14	7FFFFFA0	7		0	0	0	0
15	7FFFFFA4	7	⇓ ⇓	0	0	0	0
16	7FFFFFA8	7		1	0	0	0
17	7FFFFFAC	7	notMemS1 most significant bit	0	0	0	0
18	7FFFFFB0	8	notMemS2 least significant bit	1	0	0	1
19	7FFFFFB4	8		1	1	0	1
20	7FFFFFB8	8	⇓ ⇓	0	0	0	1
21	7FFFFFBC	8		0	0	0	0
22	7FFFFFC0	8	notMemS2 most significant bit	0	0	0	0
23	7FFFFFC4	9	notMemS3 least significant bit	1	1	1	1
24	7FFFFFC8	9		0	1	0	0
25	7FFFFFCC	9	⇓ ⇓	0	1	0	1
26	7FFFFFD0	9		0	0	1	0
27	7FFFFFD4	9	notMemS3 most significant bit	0	0	0	0
28	7FFFFFD8	10	notMemS4 least significant bit	0	0	0	1
29	7FFFFFDC	10		0	1	1	1
30	7FFFFFE0	10	⇓ ⇓	1	1	0	0
31	7FFFFFE4	10		0	0	0	0
32	7FFFFFE8	10	notMemS4 most significant bit	0	0	0	0
33	7FFFFFEC	11	Refresh Interval least significant bit	-	-	-	-
34	7FFFFFF0	11	Refresh Interval most significant bit	-	-	-	-
35	7FFFFFF4	12	Refresh Enable	-	-	-	-
36	7FFFFFF8	13	Late Write	0	1	1	0

Table 5.10 IMS T400 external configuration coding

Refresh interval	Interval in μs	Field 11 encoding	Complete cycle (ms)
18	3.6	00	0.922
36	7.2	01	1.843
54	10.8	10	2.765
72	14.4	11	3.686

Table 5.11 IMS T400 memory refresh configuration coding

Refresh intervals are in periods of **ClockIn** and **ClockIn** frequency is 5 MHz:

Interval = 18 * 200 = 3600 ns

Refresh interval is between successive incremental refresh addresses.
Complete cycles are shown for 256 row DRAMS.

Symbol	Parameter	T400-20		Units	Notes
		Min	Max		
TMCVRdH	Memory configuration data setup	25		ns	
TRdHMCX	Memory configuration data hold	0		ns	
TS0LRdH	**notMemS0** to configuration data read	388	412	ns	

Table 5.12 Memory configuration

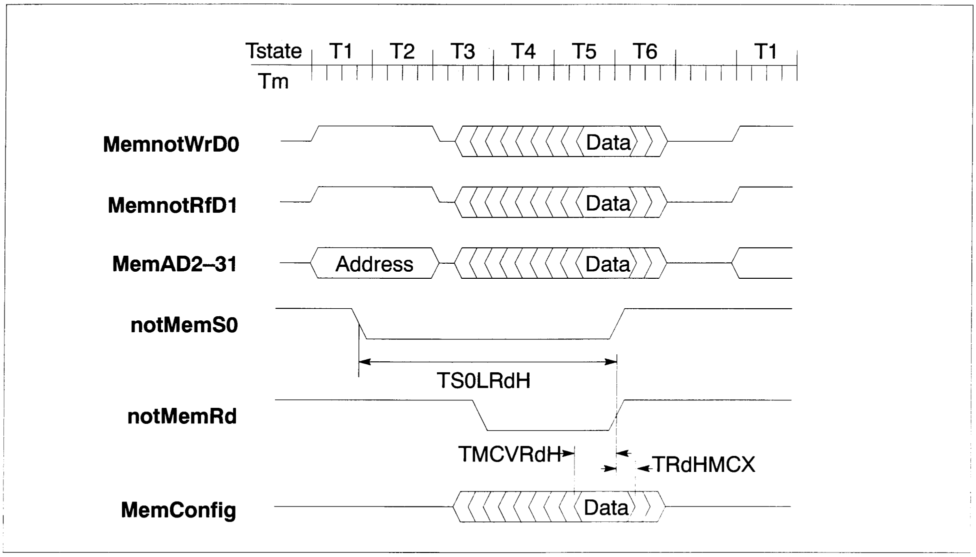


Figure 5.21 IMS T400 external configuration read cycle timing

6 Events

EventReq and **EventAck** provide an asynchronous handshake interface between an external event and an internal process. When an external event takes **EventReq** high the external event channel (additional to the external link channels) is made ready to communicate with a process. When both the event channel and the process are ready the processor takes **EventAck** high and the process, if waiting, is scheduled. **EventAck** is removed after **EventReq** goes low.

EventWaiting is asserted high by the transputer when a process executes an input on the event channel; typically with the occam `EVENT ? ANY` instruction. It remains high whilst the transputer is waiting for or servicing **EventReq** and is returned low when **EventAck** goes high. The **EventWaiting** pin changes near the falling edge of **ProcClockOut** and can therefore be sampled by the rising edge of **ProcClockOut**.

The **EventWaiting** pin can only be asserted by executing an *in* instruction on the event channel. The **EventWaiting** pin is not asserted high when an enable channel (*enbc*) instruction is executed on the Event channel (during an ALT construct in occam, for example). The **EventWaiting** pin can be asserted by executing the occam input on the event channel (such as `Event ? ANY`), provided that this does not occur as a guard in an alternative process. The **EventWaiting** pin can not be used to signify that an alternative process (ALT) is waiting on an input from the event channel.

EventWaiting allows a process to control external logic; for example, to clock a number of inputs into a memory mapped data latch so that the event request type can be determined. This function is not available on the IMS T414 and IMS T800.

Only one process may use the event channel at any given time. If no process requires an event to occur **EventAck** will never be taken high. Although **EventReq** triggers the channel on a transition from low to high, it must not be removed before **EventAck** is high. **EventReq** should be low during **Reset**; if not it will be ignored until it has gone low and returned high. **EventAck** is taken low when **Reset** occurs.

If the process is a high priority one and no other high priority process is running, the latency is as described on page 70. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

Cycle 1 Sample **EventReq** at pad on the rising edge of **ProcClockOut** and synchronise.

Cycle 2 Edge detect the synchronised **EventReq** and form the interrupt request.

Cycle 3 Sample interrupt vector for microcode ROM in the CPU.

Cycle 4 Execute the interrupt routine for Event rather than the next instruction.

Symbol	Parameter	T400–20		Units	Notes
		Min	Max		
TVHKH	**EventReq** response	0		ns	1
TKHVL	**EventReq** hold	0		ns	1
TVLKL	Delay before removal of **EventAck**	0	157	ns	
TKLVH	Delay before re-assertion of **EventReq**	0		ns	1
TKHEWL	**EventAck** to end of **EventWaiting**	0		ns	1

Notes

1 Guaranteed, but not tested.

Table 6.1 Event

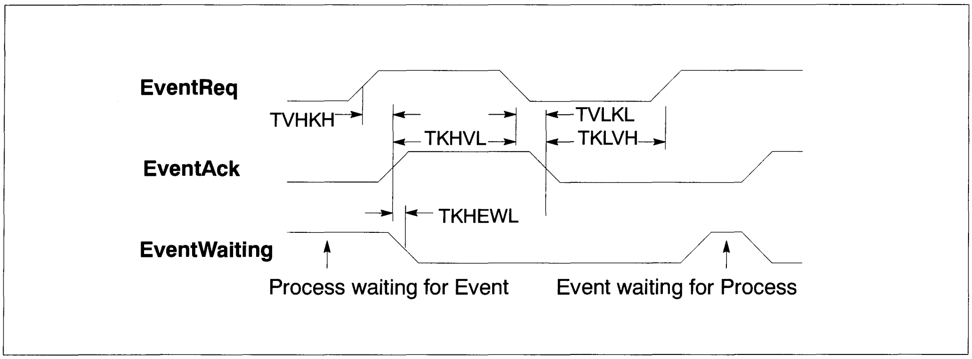


Figure 6.1 IMS T400 event timing

7 Links

Two identical INMOS bi-directional serial links provide synchronized communication between processors and with the outside world. Each link comprises an input channel and output channel. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

The IMS T400 links allow an acknowledge packet to be sent before the data packet has been fully received. This overlapped acknowledge technique is fully compatible with all other INMOS transputer links.

The IMS T400 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 5 or 20 Mbits/sec. Links are not synchronised with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

Link speeds can be set by **LinkSpecial**, **Link0Special** and **Link1Special**. Table 7.1 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed and as **Link1Special** for link 1. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

Link Special	Linkn Special	Mbits/sec	Kbytes/sec	
			Uni	Bi
0	0	10	910	1250
0	1	5	450	670
1	0	10	910	1250
1	1	20	1740	2350

Table 7.1 Speed Settings for Transputer Links

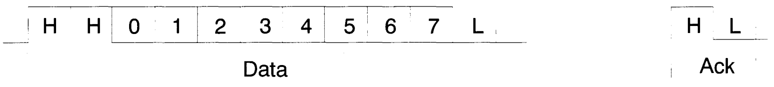


Figure 7.1 IMS T400 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	1
TJDf	**LinkIn** fall time				20	ns	1
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
		5 Mbits/s			30	ns	2
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		ohms	

Notes

1 Guaranteed, but not tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 7.2 Link

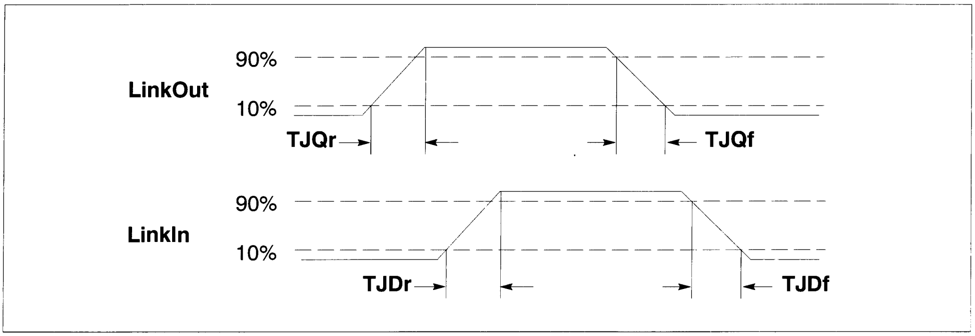


Figure 7.2 IMS T400 link timing

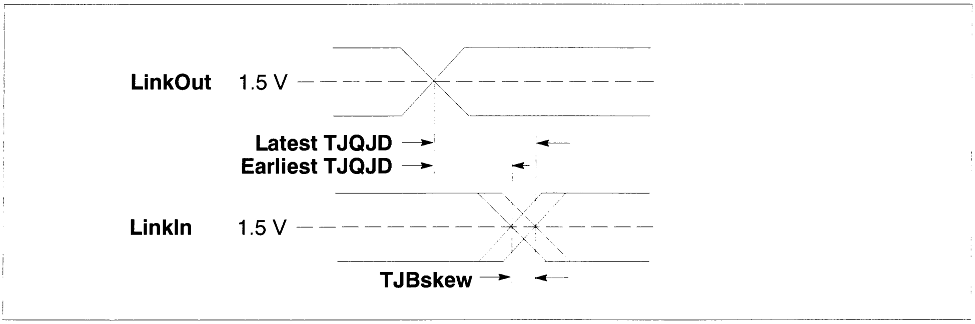


Figure 7.3 IMS T400 buffered link timing

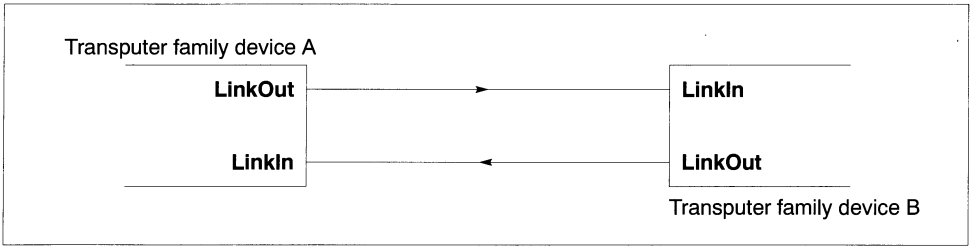


Figure 7.4 Links directly connected

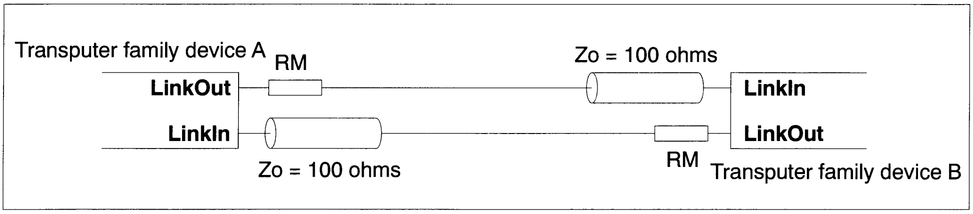


Figure 7.5 Links connected by transmission line

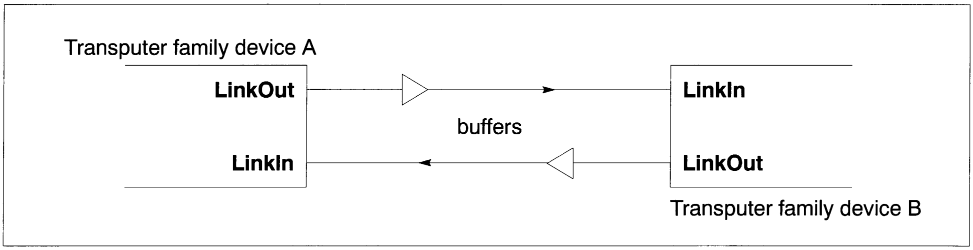


Figure 7.6 Links connected by buffers

8 Electrical specifications

8.1 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
V_I, V_O	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
I_I	Input current		±25	mA	4
t_{OSC}	Output short circuit time (one pin)		1	s	2
T_S	Storage temperature	–65	150	°C	2
T_A	Ambient temperature under bias	–55	125	°C	2
P_{Dmax}	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 8.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.5	5.5	V	1
V_I, V_O	Input or output voltage	0	VDD	V	1,2
C_L	Load capacitance on any pin		60	pF	3
T_A	Operating temperature range IMS T400-S	0	70	°C	4
T_A	Operating temperature range IMS T400-I	–40	+85	°C	4

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Excluding **LinkOut** load capacitance.

4 Air flow rate 400 linear ft/min transverse air flow.

Table 8.2 Operating conditions

SYMBOL	PARAMETER		MIN	MAX	UNITS	NOTES
VIH	High level input voltage		2.0	VDD+0.5	V	1, 2
VIL	Low level input voltage		–0.5	0.8	V	1, 2
II	Input current	@ GND<VI<VDD		±10	μA	1, 2
VOH	Output high voltage	@ IOH=2mA	VDD–1		V	1, 2
VOL	Output low voltage	@ IOL=4mA		0.4	V	1, 2
IOZ	Tristate output current	@ GND<V0<VDD		±10	μA	1, 2
PD	Power dissipation			1.0	W	2, 3
CIN	Input capacitance	@ f=1MHz		7	pF	4
COZ	Output capacitance	@ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

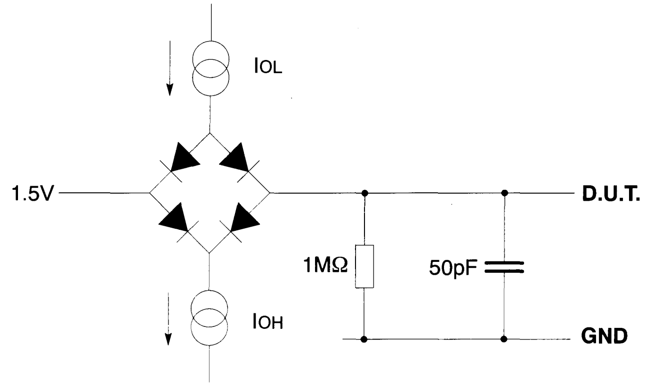
2 Parameters for IMS T400 measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C.
Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution.
Power dissipation for processor operating at 20 MHz.

4 This parameter is sampled and not 100% tested.

Table 8.3 DC characteristics

8.2 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 8.1 Load circuit for AC measurements

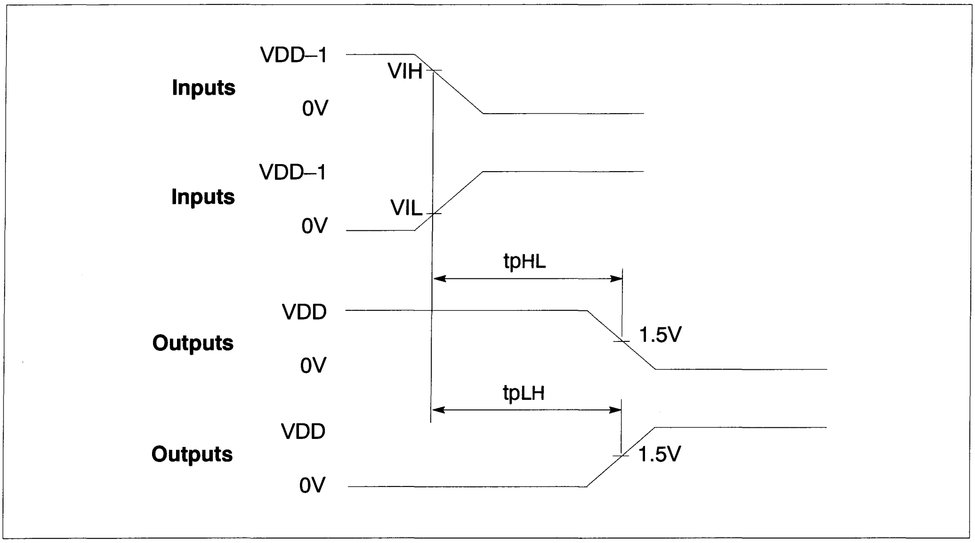


Figure 8.2 AC measurements timing waveforms

8.3 AC timing characteristics

Symbol	Parameter	Min	Max	Units	Notes
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3
TQr	Output rising edges		25	ns	1,3
TQf	Output falling edges		15	ns	1,3
TS0LaX	Address hold after **notMemS0**	**a**–8	**a**+8	ns	4

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 Guaranteed, but not tested.

4 **a** is **T2** where **T2** can be from one to four periods **Tm** in length.
Address lines include **MemnotWrD0**, **MemnotRfD1**, **MemAD2-31**.

Table 8.4 Input and output edges

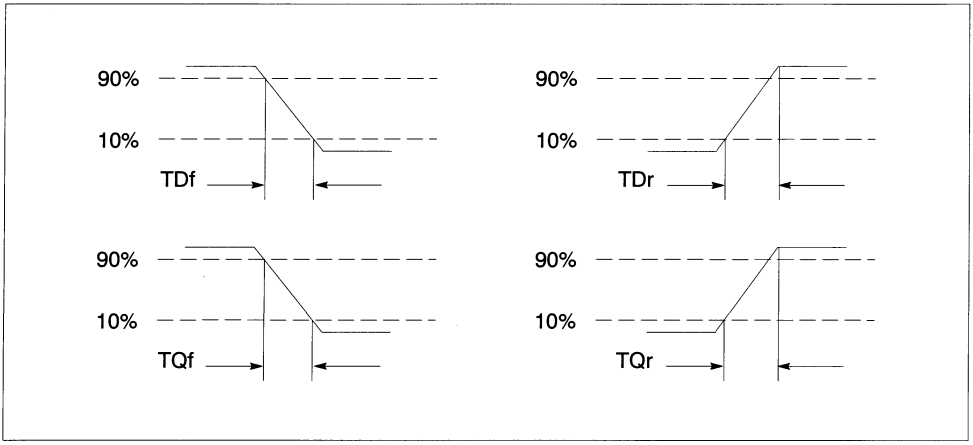


Figure 8.3 IMS T400 input and output edge timing

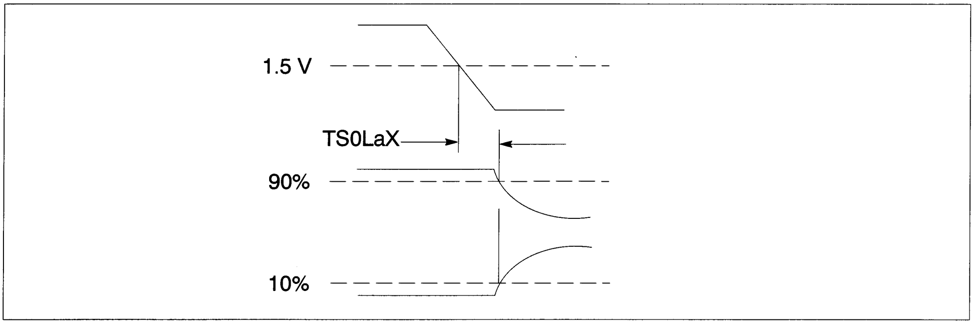
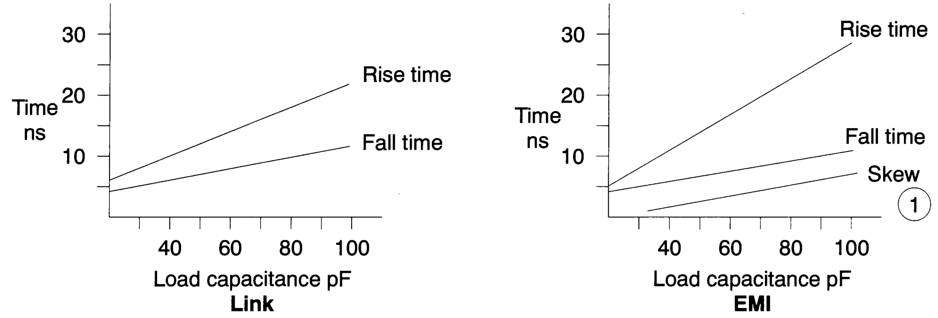


Figure 8.4 IMS T400 tristate timing relative to **notMemS0**



Notes

1 Skew is measured between **ProcClockOut** with a load of 2 Schottky TTL inputs and 30pF and **notMemS0** with a load of 2 Schottky TTL inputs and varying capacitance.

Figure 8.5 Typical rise/fall times

8.4 Power rating

Internal power dissipation (P_{INT}) of transputer and peripheral chips depends on **VDD**, as shown in figure 8.6. P_{INT} is substantially independent of temperature.

Total power dissipation (P_D) of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

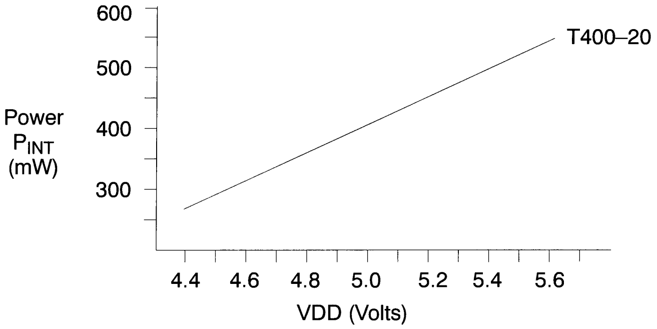


Figure 8.6 IMS T400 internal power dissipation vs VDD

9 Package pinouts

9.1 84 pin PLCC J-bend package

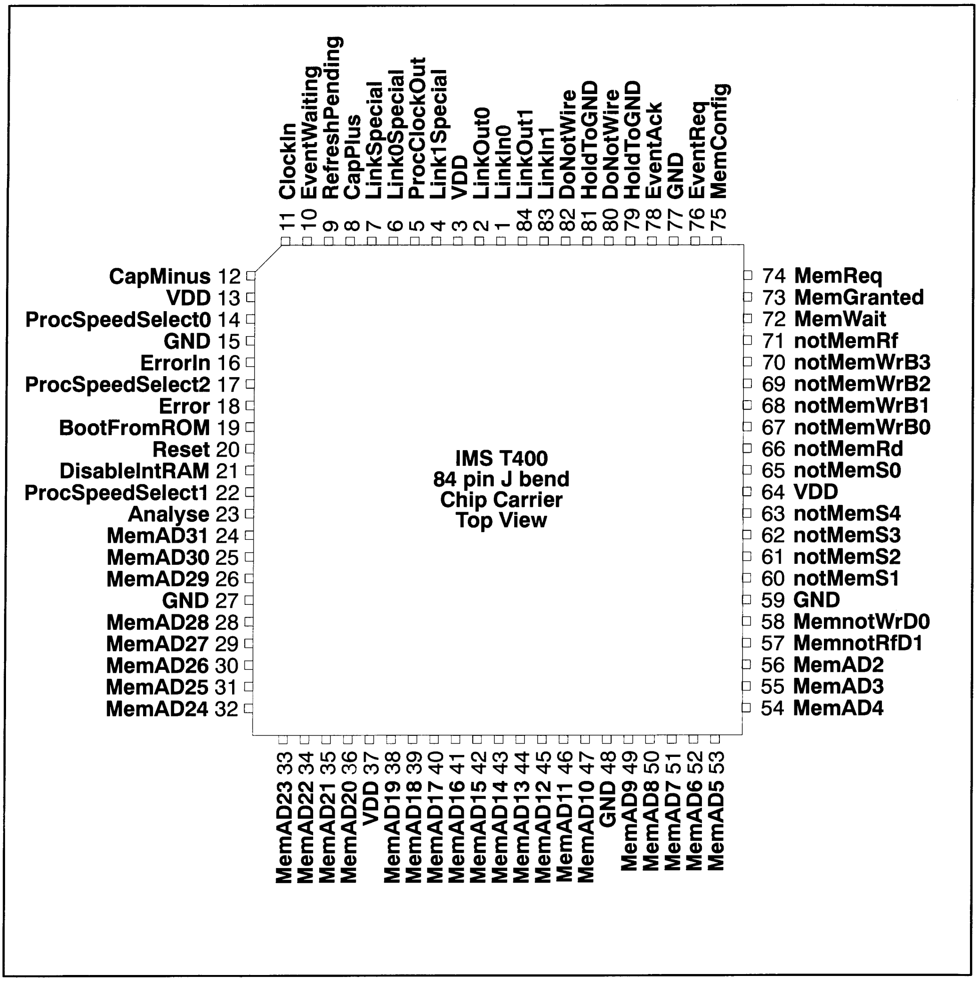
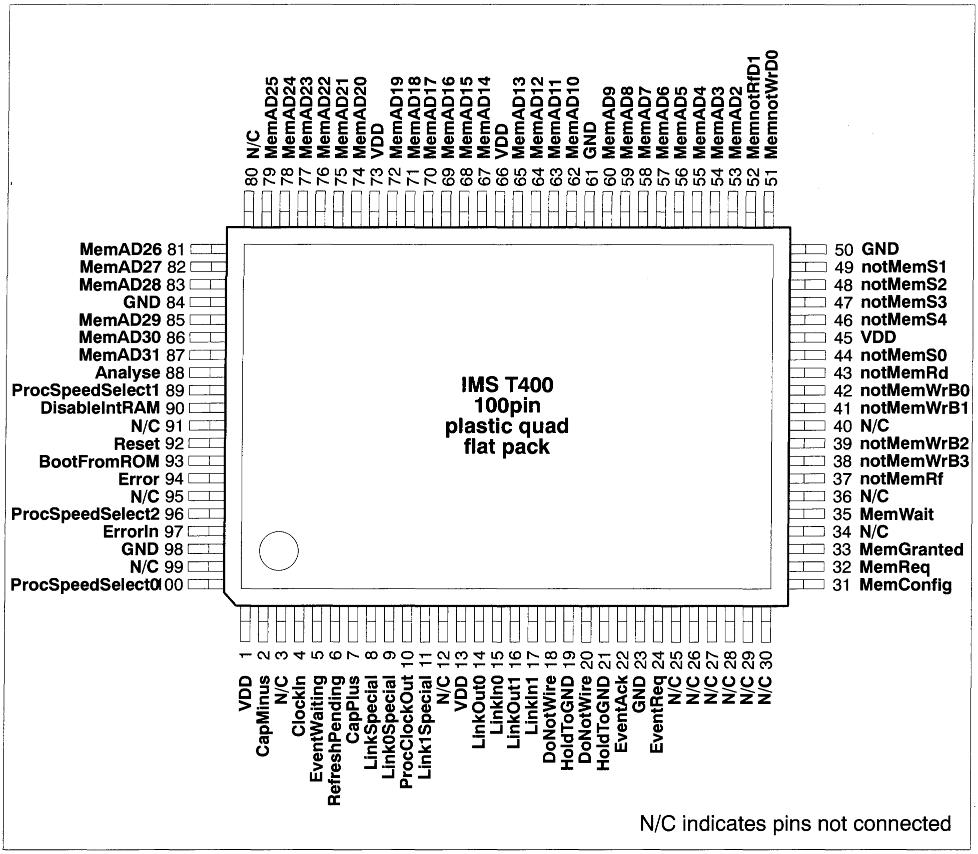


Figure 9.1 IMS T400 84 pin PLCC J-bend package pinout

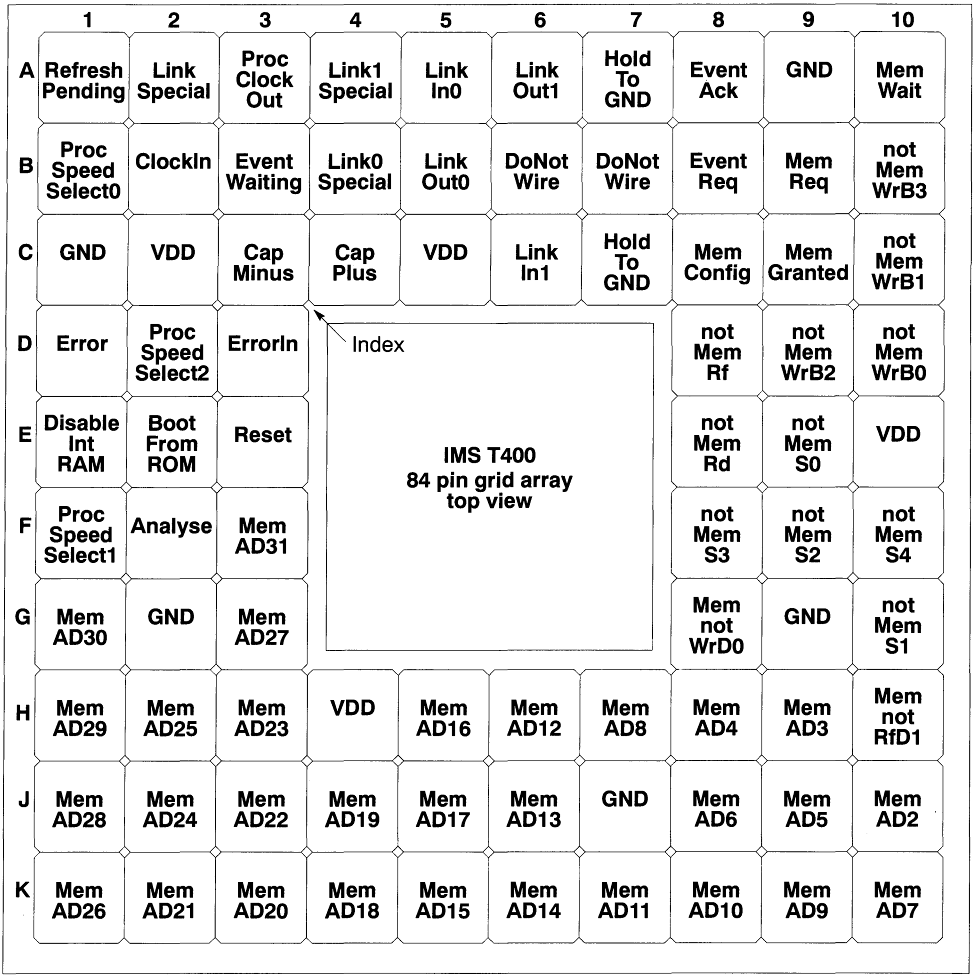
9.2 100 pin plastic quad flat pack



N/C indicates pins not connected

Figure 9.2 IMS T400 100 pin plastic quad flat pack pinout

9.3 84 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Refresh Pending	Link Special	Proc Clock Out	Link1 Special	Link In0	Link Out1	Hold To GND	Event Ack	GND	Mem Wait
B	Proc Speed Select0	ClockIn	Event Waiting	Link0 Special	Link Out0	DoNot Wire	DoNot Wire	Event Req	Mem Req	not Mem WrB3
C	GND	VDD	Cap Minus	Cap Plus	VDD	Link In1	Hold To GND	Mem Config	Mem Granted	not Mem WrB1
D	Error	Proc Speed Select2	ErrorIn					not Mem Rf	not Mem WrB2	not Mem WrB0
E	Disable Int RAM	Boot From ROM	Reset					not Mem Rd	not Mem S0	VDD
F	Proc Speed Select1	Analyse	Mem AD31					not Mem S3	not Mem S2	not Mem S4
G	Mem AD30	GND	Mem AD27					Mem not WrD0	GND	not Mem S1
H	Mem AD29	Mem AD25	Mem AD23	VDD	Mem AD16	Mem AD12	Mem AD8	Mem AD4	Mem AD3	Mem not RfD1
J	Mem AD28	Mem AD24	Mem AD22	Mem AD19	Mem AD17	Mem AD13	GND	Mem AD6	Mem AD5	Mem AD2
K	Mem AD26	Mem AD21	Mem AD20	Mem AD18	Mem AD15	Mem AD14	Mem AD11	Mem AD10	Mem AD9	Mem AD7

Figure 9.3 IMS T400 84 pin grid array package pinout

10 Ordering

This section indicates the designation of speed and package selections for the various devices. Speed of **ClockIn** is 5 MHz for all parts. Transputer processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in the external memory section.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	Processor clock speed	Processor cycle time	PLLx	Package
IMS T400-J20S	20.0 MHz	50 ns	4.0	84 pin plastic PLCC J-Bend
IMS T400-J20I	20.0 MHz	50 ns	4.0	84 pin plastic PLCC J-Bend
IMS T400-G20S	20.0 MHz	50 ns	4.0	84 pin ceramic pin grid array
IMS T400-G20I	20.0 MHz	50 ns	4.0	84 pin ceramic pin grid array
IMS T400-X20S	20.0 MHz	50 ns	4.0	100 pin plastic quad flat pack
IMS T400-X20I	20.0 MHz	50 ns	4.0	100 pin plastic quad flat pack

Table 10.1 IMS T400 ordering details

The ceramic pin grid array package option is only available in small quantities.

IMS T225 transputer

inmos®

Preliminary Data

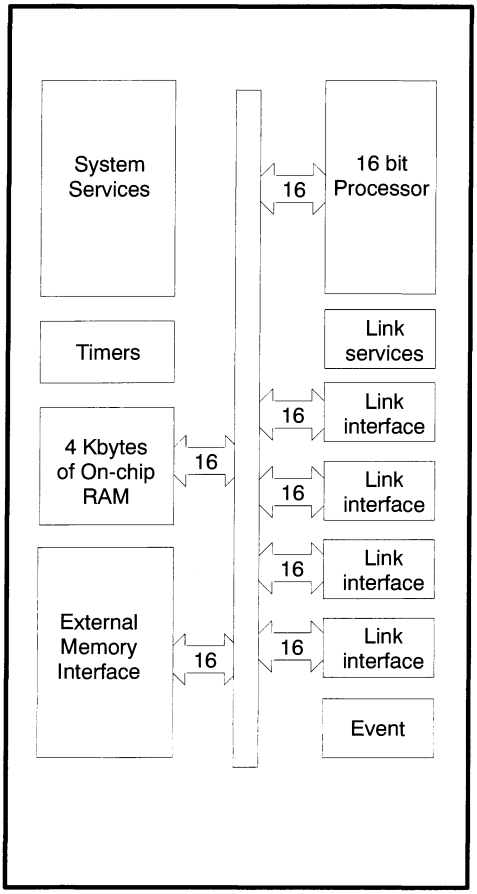
The information in this datasheet is subject to change

FEATURES

16 bit architecture
33 ns internal cycle time
30 MIPS peak instruction rate
IMS T225–20 is pin compatible with IMS T222–20
Debugging support
4 Kbytes on-chip static RAM
60 Mbytes/sec sustained data rate to internal memory
64 Kbytes directly addressable external memory
30 Mbytes/sec sustained data rate to external memory
630 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
Bi-directional data rate of 2.4 Mbytes/sec per link
Internal timers of 1μs and 64μs
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ± 5% power supply
Packaging 68 pin PGA / 68 pin PLCC / 100 pin CQFP
MIL-STD-883 processing will be available

APPLICATIONS

Real time processing
Microprocessor applications
High speed multi processor systems
Industrial control
Robotics
System simulation
Digital signal processing
Telecommunications
Fault tolerant systems
Medical instrumentation



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992

1 Introduction

The IMS T225 transputer is a 16 bit CMOS microcomputer with 4 Kbytes on-chip RAM for high speed processing, an external memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages and provides direct support for the occam model of concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond. A device running at 30 MHz achieves an instruction throughput of 15 MIPS.

For convenience of description, the IMS T225 operation is split into the basic blocks shown in figure 1.1.

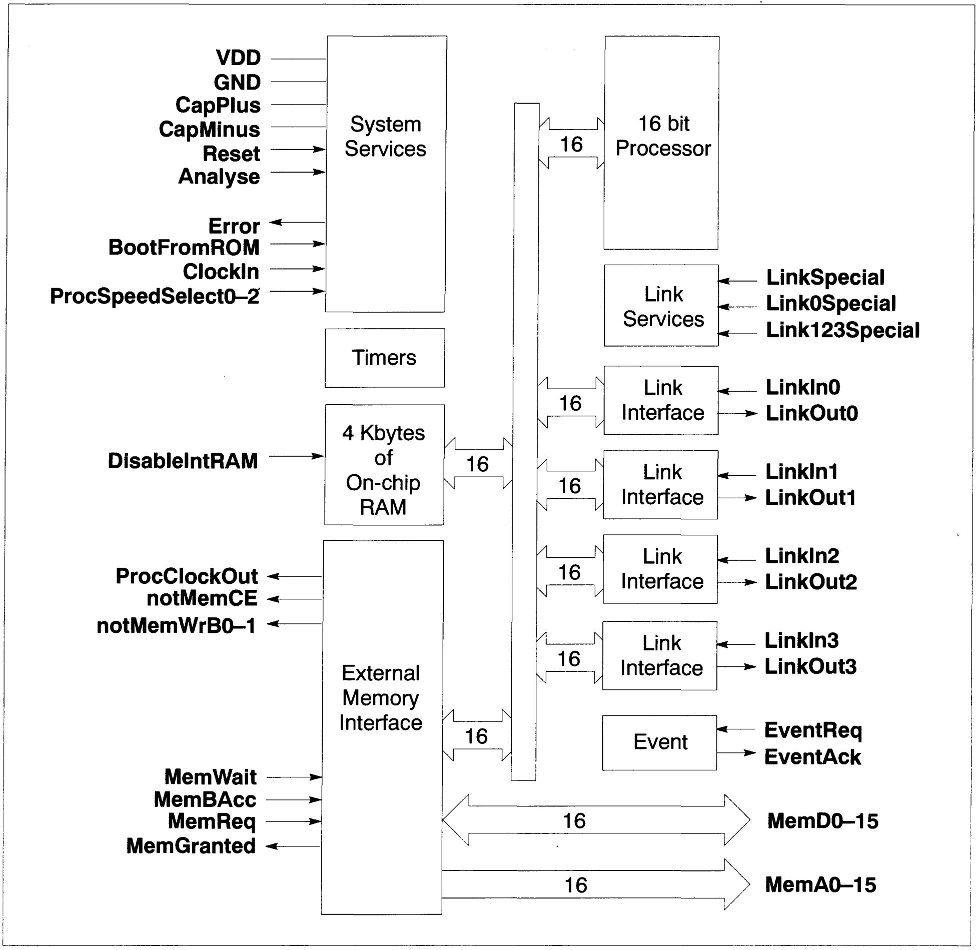


Figure 1.1 IMS T225 block diagram

The IMS T225 is functionally equivalent to the IMS T222 but has the addition of three speed select pins (**ProcSpeedSelect0-2**) and improved links. The IMS T225 is pin compatible with the IMS T222 and is a direct replacement in many applications. The IMS T225 can directly access a linear address space of 64 Kbytes. The 16 bit wide non-multiplexed external memory interface provides a data rate of up to 2 bytes every 100 nanoseconds (20 Mbytes/sec) for a 20 MHz device.

System Services include processor reset and bootstrap control, together with facilities for error analysis.

The INMOS communication links allow networks of transputers to be constructed by direct point to point connections with no external logic. The links support the standard operating speed of 10 Mbits/sec, but also operate at 5 or 20 Mbits/sec. The links have been improved over those of the IMS T222 and fully support overlapped acknowledge; each IMS T225 link can transfer data bi-directionally at up to 2.4 Mbytes/sec.

The IMS T225 is designed to implement the occam language, detailed in the *occam Reference Manual*, but also efficiently supports other languages such as C, Pascal and Fortran. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*. The instruction set of the IMS T225 is the same as that of the IMS T222 with some enhancements.

This datasheet supplies hardware implementation and characterization details for the IMS T225. It is intended to be read in conjunction with the Transputer Architecture chapter, which details the architecture of the transputer and gives an overview of occam.

The IMS T225 instruction set contains a number of instructions to facilitate the implementation of breakpoints. For further information concerning breakpointing, refer to *Support for debugging/breakpointing in transputers* (technical note 61).

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high.
Pinout details for various packages are given on page 357.

Pin	In/Out	Function
VCC, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
ProcSpeedSelect0–2	in	Processor speed selectors
Reset	in	System reset
Error	out	Error indicator
Analyse	in	Error analysis
BootFromROM	in	Boot from external ROM or from link
DisableIntRAM	in	Disable internal RAM

Table 2.1 IMS T225 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemA0–15	out	Sixteen address lines
MemD0–15	in/out	Sixteen data lines
notMemWrB0–1	out	Two byte-addressing write strobes
notMemCE	out	Chip enable
MemBAcc	in	Byte access mode selector
MemWait	in	Memory cycle extender
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted

Table 2.2 IMS T225 external memory interface

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge

Table 2.3 IMS T225 event

Pin	In/Out	Function
LinkIn0–3	in	Four serial data input channels
LinkOut0–3	out	Four serial data output channels
LinkSpecial	in	Select non-standard speed as 5 or 20 Mbits/sec
Link0Special	in	Select special speed for Link 0
Link123Special	in	Select special speed for Links 1, 2, 3

Table 2.4 IMS T225 link

3 System services

System services include all the necessary logic to initialize and sustain operation of the device. They also include error handling and analysis facilities.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimize inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1 μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 20 MHz. If a polarized capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

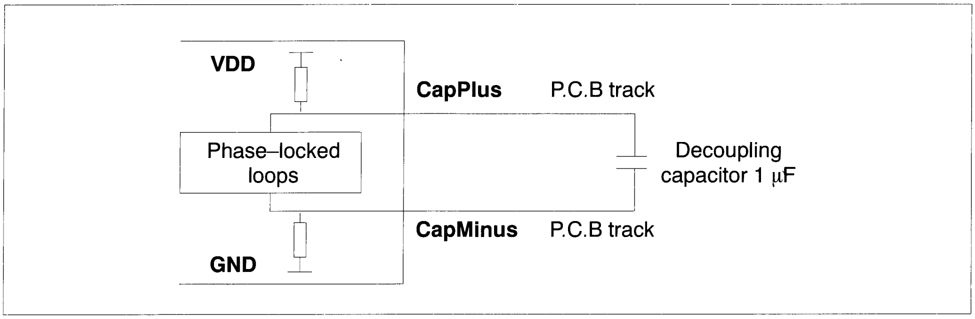


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	T225-20, -25, -30			Units	Notes
		Min	Nom	Max		
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1, 2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1, 3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1, 4
TDCr	**ClockIn** rise time			10	ns	1, 5
TDCf	**ClockIn** fall time			8	ns	1, 5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 8.3).

Table 3.1 **ClockIn** timing

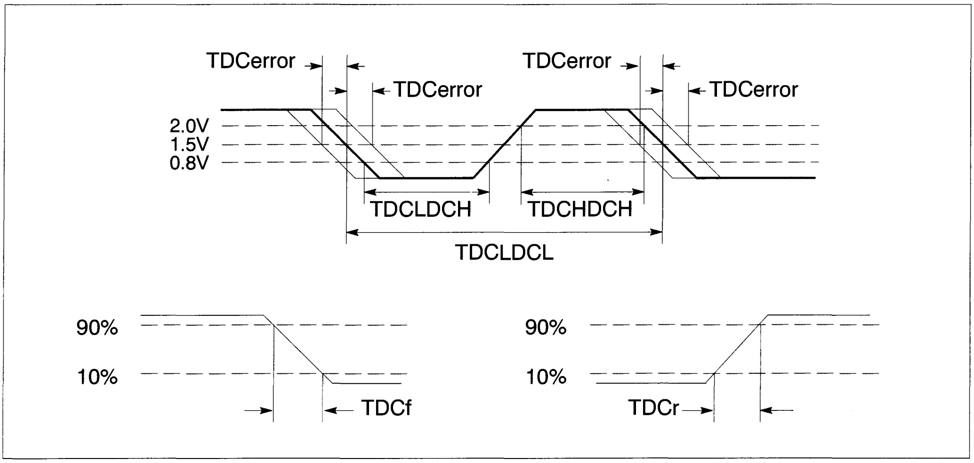


Figure 3.2 **ClockIn** timing

3.4 ProcSpeedSelect0–2

Processor speed of the IMS T225 is variable in discrete steps. The desired speed can be selected, up to the maximum rated for a particular component, by the three speed select lines **ProcSpeedSelect0-2**. The pins are tied high or low, according to the table below, for the various speeds. The pins are arranged so that the IMS T225 can be plugged directly into a board designed for a IMS T222.

Only six of the possible speed select combinations are currently used; the other two are not valid speed selectors. The frequency of **ClockIn** for the speeds given in the table is 5 MHz.

ProcSpeed-Select2	ProcSpeed-Select1	ProcSpeed-Select0	Processor Clock Speed MHz	Processor Cycle Time ns	Notes
0	0	0	20.0	50.0	
0	0	1	22.5	44.4	Not supported
0	1	0	25.0	40.0	
0	1	1	30.0	33.3	
1	0	0	35.0	28.6	Not supported
1	0	1			Invalid
1	1	0	17.5	57.1	Not supported
1	1	1			Invalid

Table 3.2 Processor speed selection

3.5 Bootstrap

The transputer can be bootstrapped either from a link or from external ROM. To facilitate debugging, **BootFromROM** may be dynamically changed but must obey the specified timing restrictions. It is sampled once only by the transputer, before the first instruction is executed after **Reset** is taken low.

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the **W** register points to *MemStart* (page 333).

If **BootFromROM** is connected low (e.g. to **GND**) the transputer will wait for the first bootstrap message to arrive on any one of its links. The transputer is ready to receive the first byte on a link within two processor cycles **TPCLPCL** after **Reset** goes low.

If the first byte received (the control byte) is greater than 1 it is taken as the quantity of bytes to be input. The following bytes, to that quantity, are then placed in internal memory starting at location *MemStart*. Following reception of the last byte the transputer will start executing code at *MemStart* as a low priority process. **BootFromROM** may be taken high after reception of the last byte, if required. The memory space immediately above the loaded code is used as work space. A byte arriving on other links after the control byte has been received and on the bootstrapping link after the last bootstrap byte, will be retained and no acknowledge will be sent until a process inputs from them.

3.6 Peek and poke

Any location in internal or external memory can be interrogated and altered when the transputer is waiting for a bootstrap from link. If the control byte is 0 then four more bytes are expected on the same link. The first two byte word is taken as an internal or external memory address at which to poke (write) the second two byte word. If the control byte is 1 the next two bytes are used as the address from which to peek (read) a word of data; the word is sent down the output channel of the same link.

Following such a peek or poke, the transputer returns to its previously held state. Any number of accesses may be made in this way until the control byte is greater than 1, when the transputer will commence reading its bootstrap program. Any link can be used, but addresses and data must be transmitted via the same link as the control byte.

3.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initializes the transputer and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) must not occur whilst **Reset** is high but can occur before bootstrap (page 345). If **BootFromROM** is high bootstrapping will take place immediately after **Reset** goes low, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

3.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 48). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags **Error** and **HaltOnError** are not altered at reset, whether **Analyse** is asserted or not.

Input links will continue with outstanding transfers. Output links will not make another access to memory for data but will transmit only those bytes already in the link buffer. Providing there is no delay in link acknowledgement, the links should be inactive within a few microseconds of the transputer halting.

Reset should not be asserted before the transputer has halted and link transfers have ceased. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 3.3.

I	**MemStart** if bootstrapping from a link, or the external memory bootstrap address if bootstrapping from ROM.
W	**MemStart** if bootstrapping from ROM, or the address of the first free word after the bootstrap program if bootstrapping from link.
A	The value of **I** when the processor halted.
B	The value of **W** when the processor halted, together with the priority of the process when the transputer was halted (i.e. the **W** descriptor).
C	The ID of the bootstrapping link if bootstrapping from link.

Table 3.3 Register values after **Analyse**

Symbol	Parameter	T225-20, -25, -30			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

1 Full periods of **ClockIn TDCLDCL** required.

2 At power-on reset.

3 Must be stable until after end of bootstrap period. See Bootstrap section 3.5.

Table 3.4 **Reset** , **Analyse** and **BootFromROM** timing

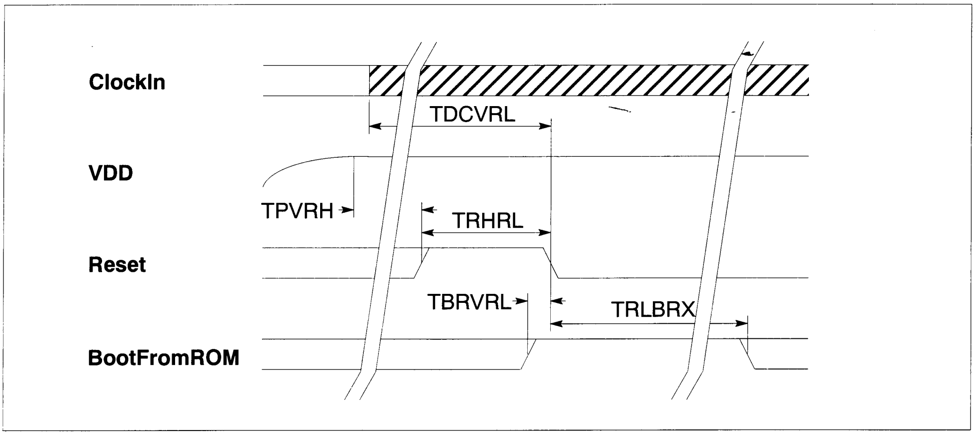


Figure 3.3 Transputer **Reset** timing with **Analyse** low

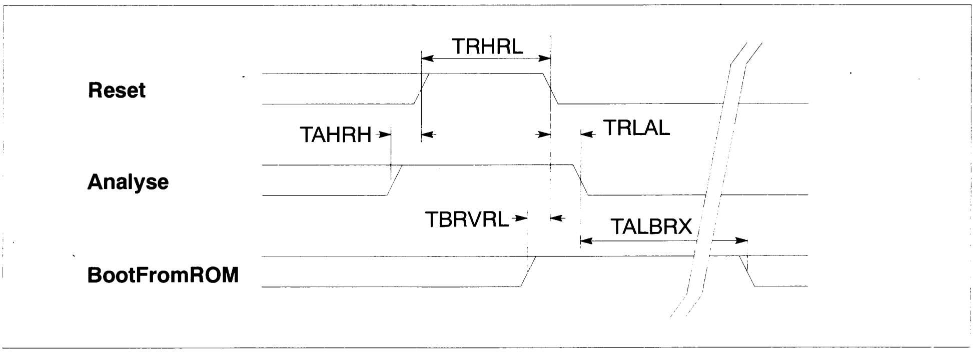


Figure 3.4 Transputer **Reset**, **Analyse** and **BootFromROM** timing

3.9 Error

The **Error** pin is connected directly to the internal *Error* flag and follows the state of that flag. If **Error** is high it indicates an error in one of the processes caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 48). Once set, the *Error* flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (page 330).

A process can be programmed to stop if the *Error* flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data.

By setting the *HaltOnError* flag the transputer itself can be programmed to halt if *Error* becomes set. If *Error* becomes set after *HaltOnError* has been set, all processes on that transputer will cease but will not necessarily cause other transputers in a network to halt. Setting *HaltOnError* after *Error* will not cause the transputer to halt; this allows the processor reset and analyse facilities to function with the flags in indeterminate states.

An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by applying the **Error** output signal of the errant transputer to the **EventReq** pin of a suitably programmed master transputer. Since the process state is preserved when stopped by an error, the master transputer can then use the analyse function to debug the fault. When using such a circuit, note that the *Error* flag is in an indeterminate state on power up; the circuit and software should be designed with this in mind.

Error checks can be removed completely to optimize the performance of a proven program; any unexpected error then occurring will have an arbitrary undefined effect.

If a high priority process pre-empts a low priority one, status of the *Error* and *HaltOnError* flags is saved for the duration of the high priority process and restored at the conclusion of it. Status of the *Error* flag is transmitted to the high priority process but the *HaltOnError* flag is cleared before the process starts. Either flag can be altered in the process without upsetting the error status of any complex operation being carried out by the pre-empted low priority process.

In the event of a transputer halting because of *HaltOnError*, the links will finish outstanding transfers before shutting down. If **Analyse** is asserted then all inputs continue but outputs will not make another access to memory for data.

After halting due to the *Error* flag changing from 0 to 1 whilst *HaltOnError* is set, register **I** points two bytes past the instruction which set *Error*. After halting due to the **Analyse** pin being taken high, register **I** points one byte past the instruction being executed. In both cases **I** will be copied to register **A**.

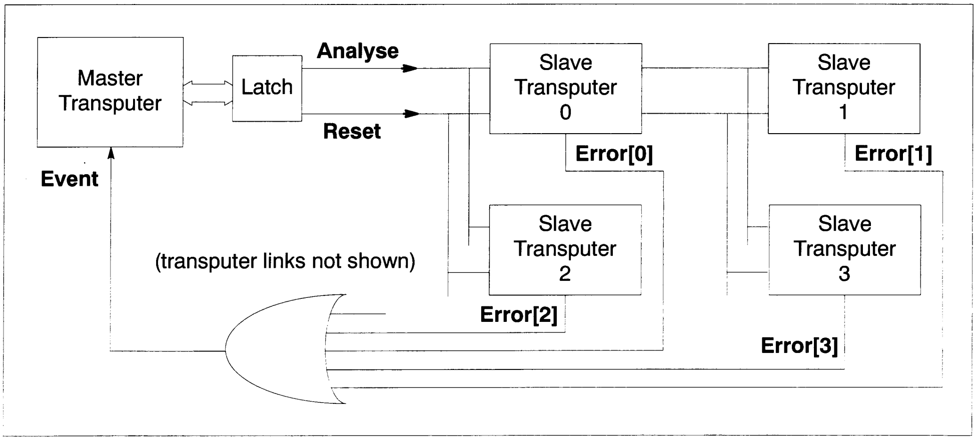


Figure 3.5 Error handling in a multi-transputer system

4 Memory

The IMS T225 has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut**. The transputer can also access an additional 60 Kbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

IMS T225 memory is byte addressed, with words aligned on two-byte boundaries. The least significant byte of a word is the lowest addressed byte.

The bits in a byte are numbered 0 to 7, with bit 0 the least significant. The bytes are numbered from 0, with byte 0 the least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

Internal memory starts at the most negative address #8000 and extends to #8FFF. User memory begins at #8024; this location is given the name *MemStart*. An instruction *ldmemstartval* is provided to obtain the value of **MemStart**.

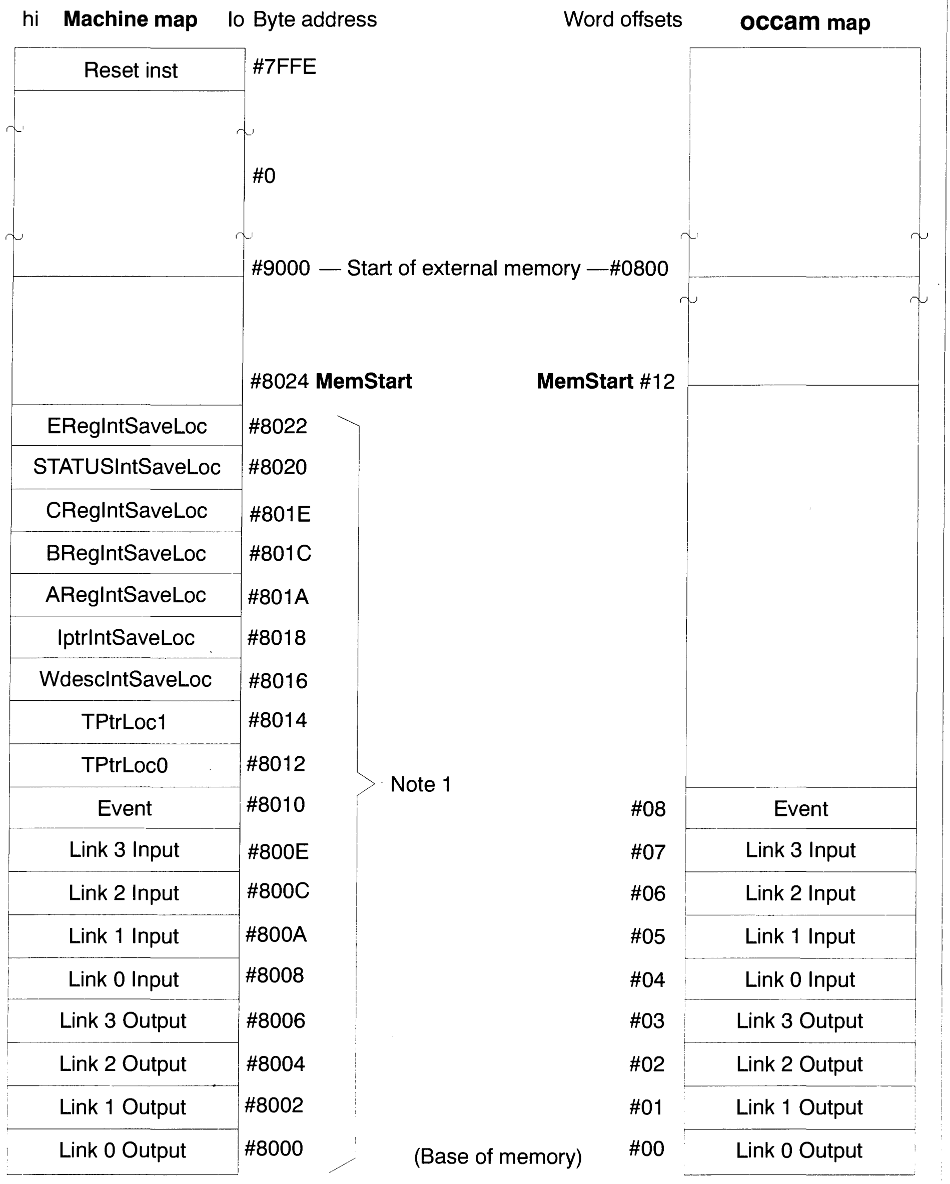
The context of a process in the transputer model involves a workspace descriptor (**WPtr**) and an instruction pointer (**IPtr**). **WPtr** is a word address pointer to a workspace in memory. **IPtr** points to the next instruction to be executed for the process which is the currently executing process. The context switch performed by the breakpoint instruction swaps the **WPtr** and **IPtr** of the currently executing process with the **WPtr** and **IPtr** held above **MemStart**. Two contexts are held above **MemStart**, one for high priority and one for low priority; this allows processes at both levels to have breakpoints. Note that on bootstrapping from a link, these contexts are overwritten by the loaded code. If this is not acceptable, the values should be peeked from memory before bootstrapping from a link.

The reserved area of internal memory below **MemStart** is used to implement link and event channels.

Two words of memory are reserved for timer use, *TPtrLoc0* for high priority processes and *TPtrLoc1* for low priority processes. They either indicate the relevant priority timer is not in use or point to the first process on the timer queue at that priority level.

Values of certain processor registers for the current low priority process are saved in the reserved *IntSaveLoc* locations when a high priority process pre-empts a low priority one.

External memory space starts at #9000 and extends up through #0000 to #7FFF. ROM bootstrapping code must be in the most positive address space, starting at #7FFE. Address space immediately below this is conventionally used for ROM based code.



Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 330). For details see the *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 4.1 IMS T225 memory map

5 External memory interface

The IMS T225 External Memory Interface (EMI) can access a 64 Kbyte physical address space, and provides a sustained bandwidth of 30 Mbytes/sec. It accesses a 16 bit wide address space via separate address and data buses. The data bus can be configured for either 16 bit or 8 bit memory access, allowing the use of a single bank of byte-wide memory. Both word-wide and byte-wide access may be mixed in a single memory system (see section 5.5).

The timing parameters given in this chapter are based on tests on a limited number of samples and may change when full characterization is completed.

The external memory cycle is divided into four **Tstates** with the following functions:

T1 Address and control setup time.

T2 Data setup time.

T3 Data read/write.

T4 Data and address hold after access.

Each **Tstate** is half a processor cycle **TPCLPCL** long (see section 5.2). An external memory cycle is always a complete number of cycles **TPCLPCL** in length and the start of **T1** always coincides with a rising edge of **ProcClockOut**. **T2** can be extended indefinitely by adding externally generated wait states of one complete processor cycle each.

During an internal memory access cycle the external memory interface address bus **MemA0-15** reflects the word address used to access internal RAM, **notMemWrB0-1** and **notMemCE** are inactive and the data bus **MemD0-15** is tristated. This is true unless and until a DMA (memory request) activity takes place, when the lines will be placed in a high impedance state by the transputer.

Bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 330).

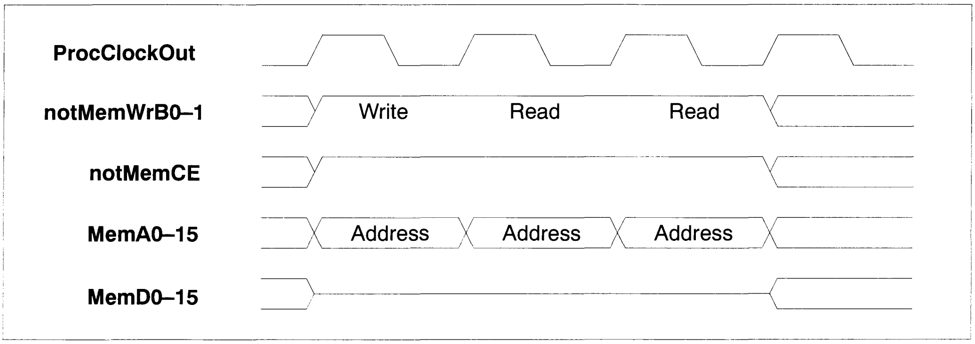


Figure 5.1 IMS T225 bus activity for 3 internal memory cycles

Figure 5.2 below shows an example of an IMS T225 being used in a static RAM application.

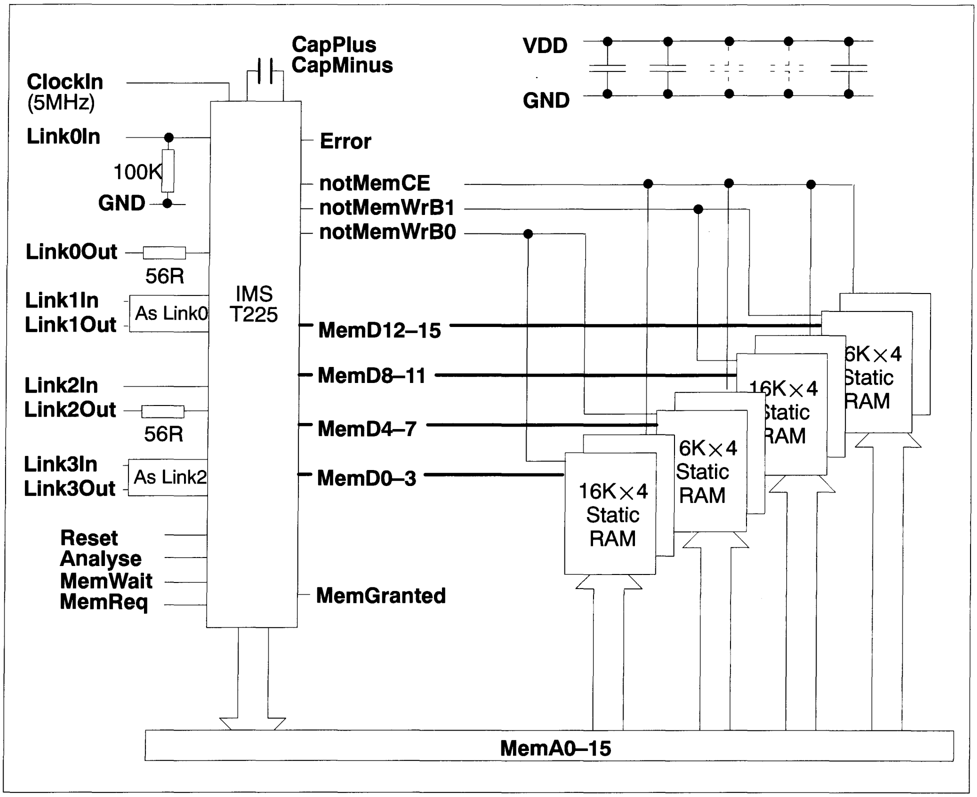


Figure 5.2 IMS T225 static RAM application

5.1 Pin functions

5.1.1 MemA0–15

External memory addresses are output on a non-multiplexed 16 bit address bus (**MemA0–15**). The address is valid at the start of **T1** and remains so until the end of **T4**. Byte addressing is carried out internally by the IMS T225 for read cycles. For write cycles the relevant bytes in memory are addressed by the write enables **notMemWrB0-1**.

5.1.2 MemD0–15

The non-multiplexed data bus (**MemD0–15**) is 16 bits wide. Read cycle data may be set up on the bus at any time after the start of **T1**, but must be valid when the IMS T225 reads it during **T4**. Data can be removed any time after the rising edge of **notMemCE**, but must be off the bus no later than the middle of **T1**, which allows for bus turn-around time before the data lines are driven at the start of **T2** in a processor write cycle. Write data is placed on the bus at the start of **T2** and removed at the end of **T4**. The writing of data into memory is normally synchronized to **notMemCE** going high.

The data bus is high impedance except when the transputer is writing data. If only one byte is being written, the unused 8 bits of the bus are high impedance at that time.

5.1.3 notMemWrB0–1

Two write enables are provided, one to write each byte of the word. When writing a word, both write enables are asserted; when writing a byte only the appropriate write enable is asserted. **notMemWrB0** addresses the least significant byte.

The write enables are synchronized with the chip enable signal **notMemCE**, allowing them to be used without **notMemCE** for simple designs.

Data may be strobed into memory using **notMemWrB0-1** without the use of **notMemCE**, as the write enables go high between consecutive external memory write cycles. The write enables are placed in a high impedance state during DMA, and are inactive during internal memory access.

5.1.4 notMemCE

The active low signal **notMemCE** is used to enable external memory on both read and write cycles.

5.1.5 MemBAcc

The IMS T225 performs word access at even memory locations. Access to byte-wide memory can be achieved by taking **MemBAcc** high. Where all external memory operations are to byte-wide memory, **MemBAcc** may be wired permanently high. The state of this signal is latched during **T2**.

If **MemBAcc** is low then a full word will be accessed in one external memory cycle, otherwise the high and low bytes of the word will be separately accessed during two consecutive cycles. The first (least significant) byte is accessed at the word address (**MemA0** is low). The second (most significant) byte is accessed at the word address +1 (**MemA0** is high).

5.1.6 MemWait

If the data setup time for read or write is too short it can be extended by inserting wait states at the end of **T2** (see section 5.6). Wait states can be selected by taking **MemWait** high. **MemWait** is sampled during **T2**, and should not change state in this region.

Internal memory access is unaffected by the number of wait states selected.

5.1.7 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. **MemGranted** can be used to signal to the device requesting the DMA that it has control of the bus.

For external memory cycles, the IMS T225 samples **MemReq** during the first high phase of **ProcClockOut** after **notMemCE** goes low. In the absence of an external memory cycle, **MemReq** is sampled during every rising edge of **ProcClockOut**. **MemA0-15**, **MemD0-15**, **notMemWrB0-1** and **notMemCE** are tristated before **MemGranted** is asserted.

5.1.8 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn** (see section 5.2).

5.2 Processor clock

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn**. Its period is equal to one internal microcode cycle time, and can be derived from the formula

TPCLPCL = TDCLDCL / PLLx

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (refer to chapter 10).

Edges of the various external memory strobes are synchronized by, but do not all coincide with, rising or falling edges of **ProcClockOut**.

		T225-20		**T225-25**		**T225-30**			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TPCLPCL	**ProcClockOut** period	48	52	38	42	31	35	ns	2
TPCHPCL	**ProcClockOut** pulse width high	18	32	14	26	13	22	ns	2
TPCLPCH	**ProcClockOut** pulse width low	**a**		**a**		**a**		ns	2, 3, 4
TPCstab	**ProcClockOut** stability		8		8		8	%	1, 2

Notes

1 Stability is the variation of cycle periods between two consecutive cycles, measured at corresponding points on the cycles.

2 This parameter is sampled and not 100% tested.

3 **a** is TPCLPCL – TPCHPCL.

4 This is a nominal value.

Table 5.1 **ProcClockOut**

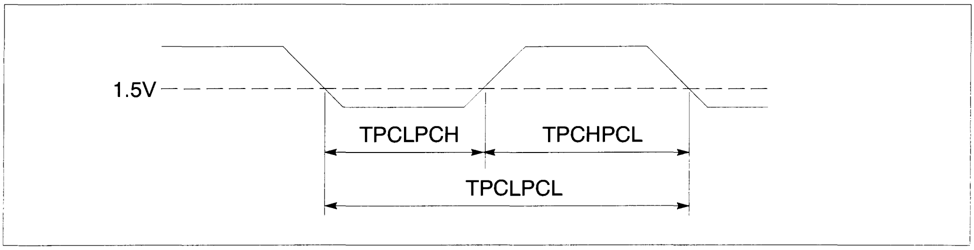


Figure 5.3 IMS T225 **ProcClockOut** timing

5.3 Read cycles

External memory addresses are output on the non-multiplexed 16 bit address bus (**MemA0–15**). The address is valid at the start of **T1** and remains so until the end of **T4**, with the timing shown in figure 5.4. Byte addressing is carried out internally by the IMS T225 for read cycles.

The non-multiplexed data bus (**MemD0–15**) is 16 bits wide. Read cycle data may be set up on the data bus at any time after the start of **T1**, but must be valid when the IMS T225 reads it during **T4**. Data can be removed any time after the rising edge of **notMemCE**, but must be off the bus no later than the middle of **T1**, which allows for bus turn-around time before the data lines are driven at the start of **T2** in a processor write cycle.

Symbol	Parameter	T225-20		T225-25		T225-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TAVEL	Address valid before chip enable low	8		6		4		ns	1
TELEH	Chip enable low	68	80	54	66	44	56	ns	1
TEHEL	Delay before chip enable re-assertion	19		15		11		ns	1, 2
TEHAX	Address hold after chip enable high	3		1		1		ns	1
TELDrV	Data valid from chip enable low	0	50	0	40	0	33	ns	
TAVDrV	Data valid from address valid	0	63	0	53	0	45	ns	
TDrVEH	Data setup before chip enable high	22		17		13		ns	
TEHDrZ	Data hold after chip enable high	0		0		0		ns	
TWEHEL	Write enable setup before chip enable low	18		16		13		ns	3
TPCHEL	**ProcClockOut** high to chip enable low	8	20	4	17	2	13	ns	1
TEHPCH	Chip enable high to **ProcClockOut** high	5		5		5		ns	

Notes

1 This parameter is common to read and write cycles and to byte-wide memory accesses.

2 These values assume back-to-back external memory accesses.

3 Timing is for both write strobes **notMemWrB0-1**.

Table 5.2 Read

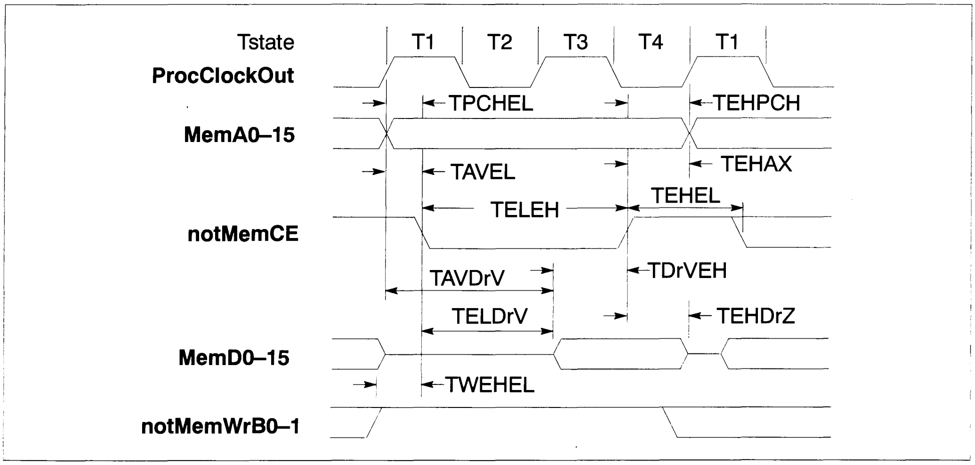


Figure 5.4 IMS T225 external read cycle

5.4 Write cycles

For write cycles the relevant bytes in memory are addressed by the write strobes **notMemWrB0-1**.

Write data is placed on the data bus (**MemD0–15**) at the start of **T2** and removed at the end of **T4**. It is normally written into memory in synchronism with **notMemCE** going high.

Two write strobes are provided, one to write each byte of the word. When writing a word, both write strobes are asserted; when writing a byte only the appropriate write enable is asserted. **notMemWrB0** addresses the least significant byte.

The IMS T225 will, by default, perform word access at even memory locations, this is termed word access mode. Access to byte-wide memory can be achieved by taking the **MemBAcc** signal high, this is termed byte access mode (see section 5.5.2). In word access mode a full word will be accessed in one external memory cycle, in byte access mode the high and low bytes of the word will be separately accessed during two consecutive cycles. Both word-wide and byte-wide access may be mixed in a single memory system. Figure 5.6 shows a write access of the least significant byte of the word when in word access mode (**MemBAcc** low). **MemA0** is low to signify access of the least significant byte of the word at the word address.

Data may be strobed into memory using **notMemWrB0-1** without the use of **notMemCE**, as the write strobes go high between consecutive external memory write cycles. The write strobes are placed in a high impedance state during DMA, and are inactive during internal memory access.

		T225-20		T225-25		T225-30			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TDwVEH	Data setup before chip enable high	50		40		33		ns	
TEHDwZ	Data hold after write	5	25	3	20	3	16.5	ns	
TDwZEL	Write data invalid to next chip enable	1		1		1		ns	
TWELEL	Write enable setup before chip enable low	–8	3	–3	3	–3	3	ns	1
TEHWEH	Write enable hold after chip enable high	–3	6	–3	3	–3	3	ns	1

Notes

1 Timing is for both write strobes **notMemWrB0-1**.

Table 5.3 Write

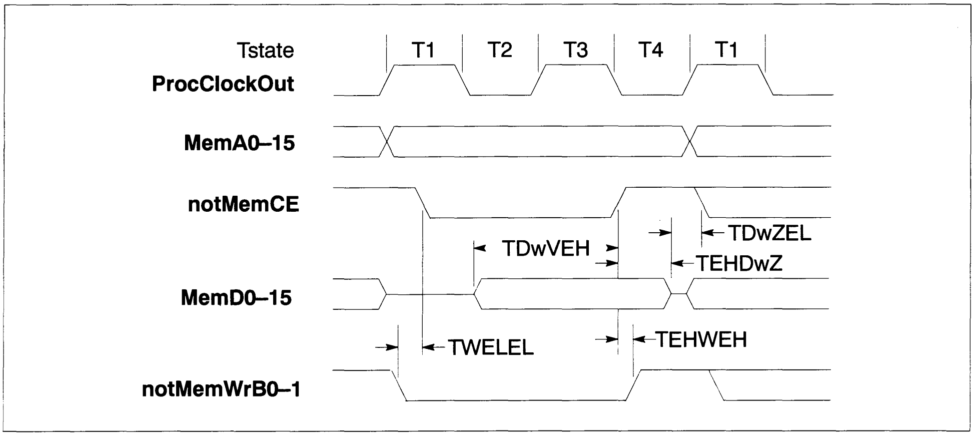


Figure 5.5 IMS T225 external write cycle

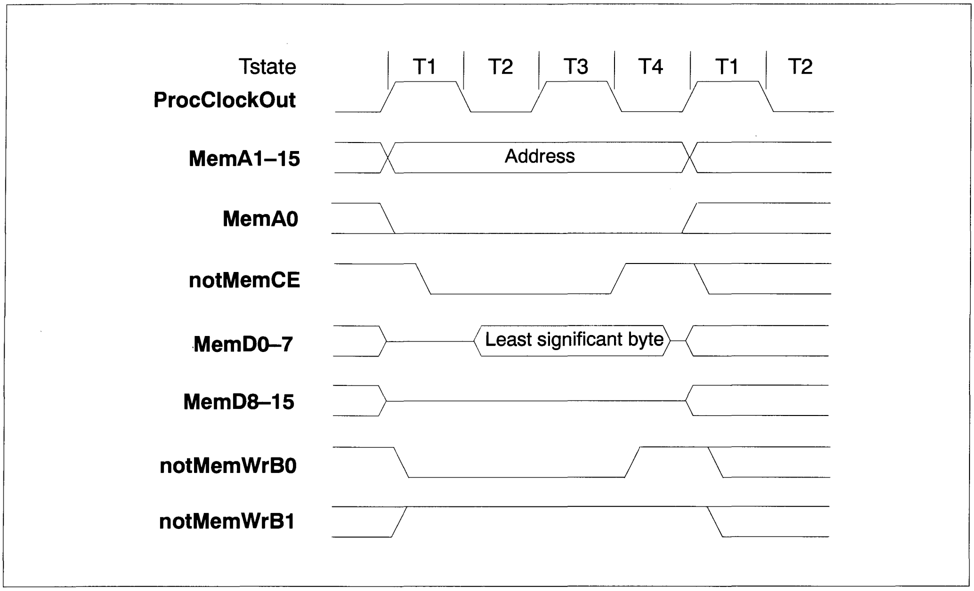


Figure 5.6 IMS T225 least significant byte write in word access mode

5.5 MemBAcc

The IMS T225 will, by default, perform word access at even memory locations. Access to byte-wide memory can be achieved by taking **MemBAcc** high with the timing shown. Where all external memory operations are to byte-wide memory, **MemBAcc** may be wired permanently high. The state of this signal is latched during **T2**. Where external memory operations may be to both byte and word wide memory, **MemBAcc** should be obtained by address decoding. If you use a memory system in which word wide memory may be used in byte access mode it is recommended that **notMemWrB1** is OR gated with **MemA0**, to prevent any spurious data being written to the RAM.

If **MemBAcc** is low then a full word will be accessed in one external memory cycle, otherwise the high and low bytes of the word will be separately accessed during two consecutive cycles. The first (least significant) byte is accessed at the word address (**MemA0** is low). The second (most significant) byte is accessed at the word address +1 (**MemA0** is high).

5.5.1 Word Read/Write in Byte Access Mode

With **MemBAcc** high, the first cycle is identical with a normal word access cycle. However, it will be immediately followed by another memory cycle, which will use **MemD0-7** to read or write the second (most significant) byte of data. During this second cycle, for a write, **notMemWrB0–1** both go low as in the first cycle and **MemA0** goes high. For a read, **notMemWrB0–1** remain high and **MemA0** goes high. **MemD8–15** are high impedance for both read and write in the second cycle. Figure 5.7 shows a word write to byte-wide memory.

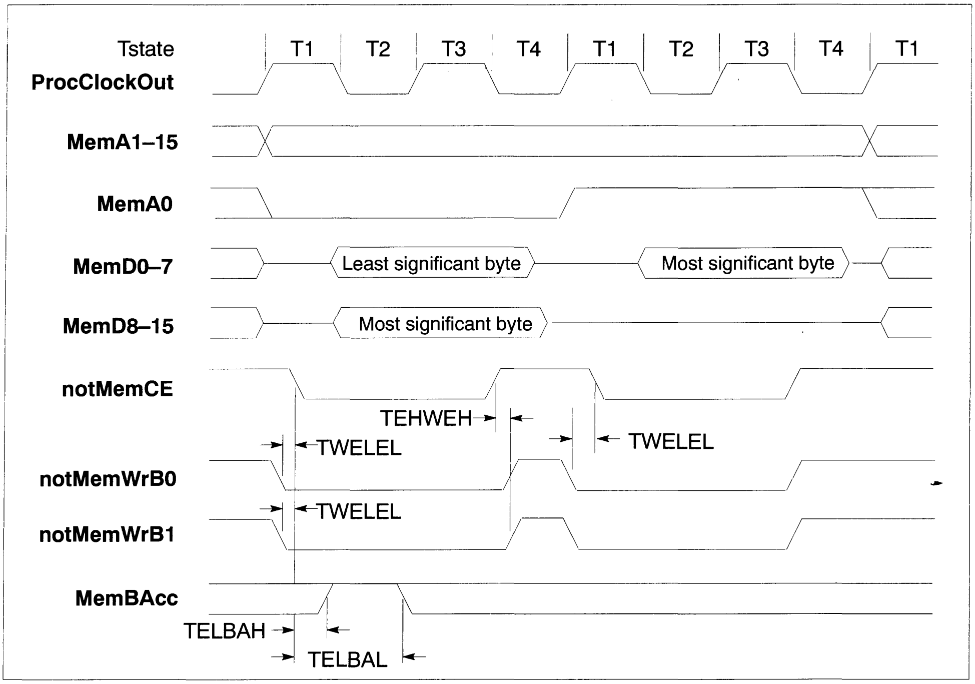


Figure 5.7 IMS T225 word write to byte-wide memory

5.5.2 Byte Write in Byte Access Mode

Writing a Most Significant Byte

In the first cycle **notMemWrB1** will go low and **notMemWrB0** will remain high. **MemA0** remains low. In the second cycle **MemA0** goes high and **notMemWrB0–1** go low. The data is written on **MemD0–7** in the second cycle.

Figure 5.8 shows a write access of the most significant byte of the word when in byte access mode (**MemBAcc** high). During the first access a normal word access is performed with **notMemWrB1** active low to select the most significant byte. During the second cycle, **MemD0-7** writes the most significant byte accessed at word address + 1 (**MemA0** is high).

Writing a Least Significant Byte

In the first cycle **notMemWrB1** remains high and **notMemWrB0** goes low. **MemA0** remains low. In the second cycle **MemA0** and **notMemWrB0** go high, **notMemWrB1** remains high. Data is written on **MemD0–7** in the first cycle.

		T225-20		T225-25		T225-30			
Symbol	Parameter	Min	Max	Min	Max	Min	Max	Units	Notes
TELBAH	**MemBAcc** high from chip enable		12		10		8	ns	
TELBAL	**MemBAcc** low from chip enable	32		27		23		ns	

Table 5.4 Byte-wide memory access

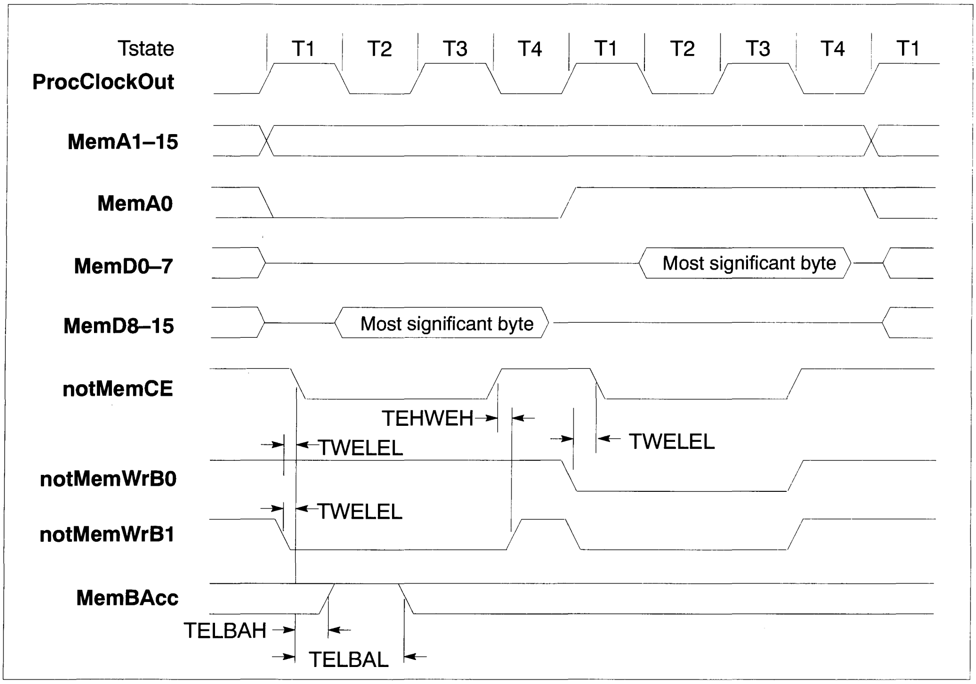


Figure 5.8 IMS T225 most significant byte write to byte-wide memory

5.6 Wait

Wait states can be selected by taking **MemWait** high. **MemWait** is sampled during **T2**, and should not change state in this region.

A wait state is one processor cycle TPCLPCL long and comprises the pair **W1** and **W2,** each half a processor cycle long (see figure 5.9). If **MemWait** is still high when sampled in **W2** then another wait period will be inserted. This can continue indefinitely. Internal memory access is unaffected by the number of wait states selected.

The setup and hold timing requirements for **MemWait** are the same as for a normal word read/write cycle. Each wait state inserted extends the length of **MemA0-15**, **MemD0-7**, **notMemCE**, and **notMemWrB0** by one **ProcClockOut** cycle (TPCLPCL).

If wait states are required to extend a byte access cycle, then the time in the cycle at which **MemBAcc** needs to be asserted is delayed (relative to the falling edge of **notMemCE**) by one TPCLPCL for each wait state inserted (see figure 7.9). **MemWait** also needs to be re-asserted for the second byte accessed (**MemA0** is high), to extend this cycle. The timing requirements of the second assertion of wait are identical to the first except that the timing is relative to the equivalent rising edge of **ProcClockOut** in the second byte access. Note that the number of wait states inserted in the even and odd byte accesses can be different.

Symbol	Parameter	T225-20		T225-25		T225-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TPCHWtH	**MemWait** asserted after **ProcClockOut** high		25		20		16.5	ns	
TPCHWtL	**MemWait** low after **ProcClockOut** high	45		40		33		ns	

Table 5.5 Memory wait

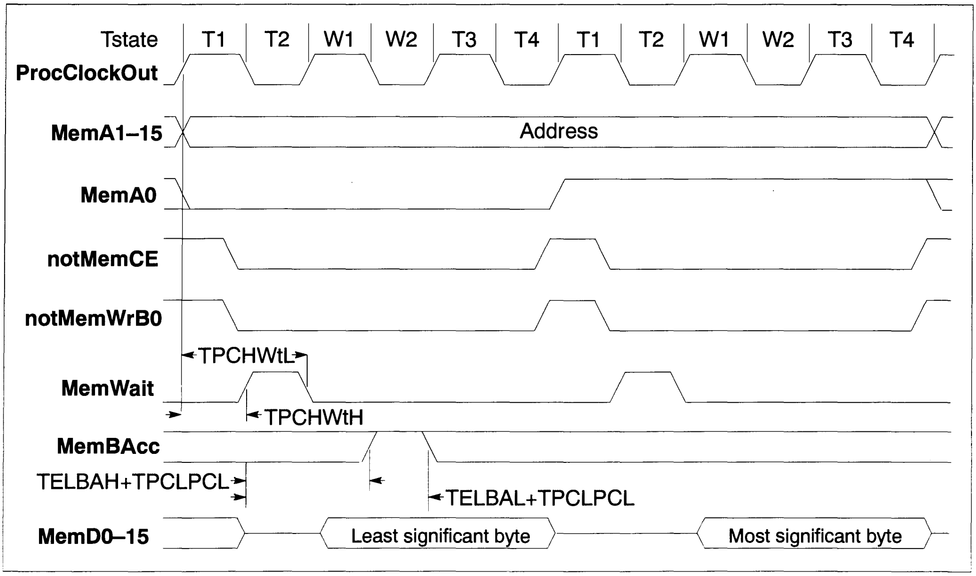


Figure 5.9 IMS T225 word write to byte wide memory with a single wait state

The wait state generator can be a simple digital delay line, synchronized to **notMemCE**. The **Single Wait State Generator** circuit in figure 5.10 can be extended to provide two or more wait states, as shown in figure 5.11.

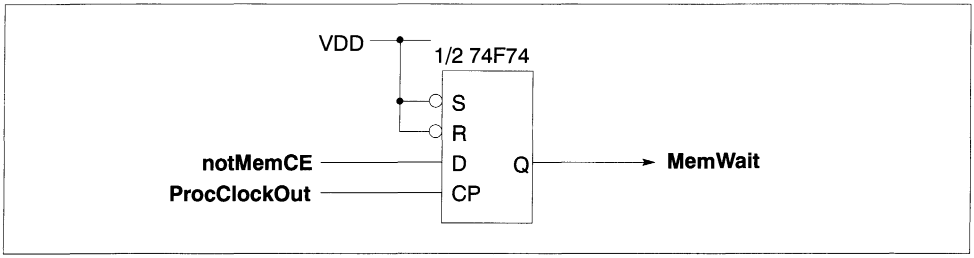


Figure 5.10 Single wait state generator

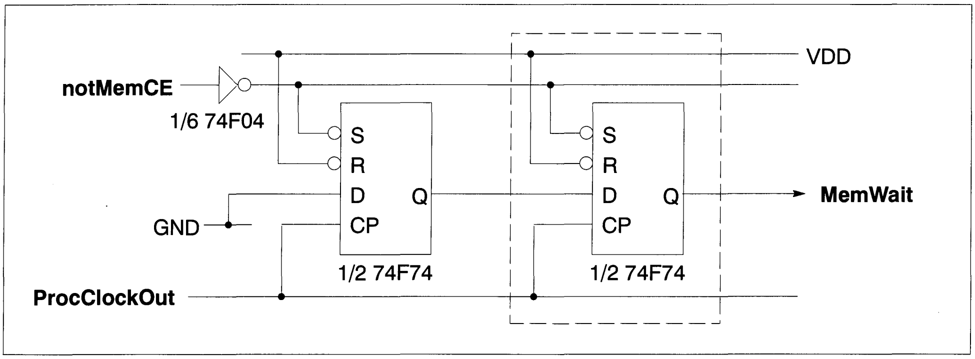


Figure 5.11 Extendable wait state generator

5.7 Direct memory access

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. For external memory cycles, the IMS T225 samples **MemReq** during the first high phase of **ProcClockOut** after **notMemCE** goes low. In the absence of an external memory cycle, **MemReq** is sampled during every rising edge of **ProcClockOut**. **MemA0-15**, **MemD0-15**, **notMemWrB0-1** and **notMemCE** are tristated before **MemGranted** is asserted.

Removal of **MemReq** is sampled at each rising edge of **ProcClockOut** and **MemGranted** removed with the timing shown in figure 5.14. Further external bus activity, either external cycles or reflection of internal cycles, will commence during the next low phase of **ProcClockOut**.

notMemCE, **notMemWrB0-1**, **MemA0-15** and **MemD0-15** are in a high impedance state during DMA. External circuitry must ensure that **notMemCE** and **notMemWrB0-1** do not become active whilst control is being transferred; it is recommended that a 10K resistor is connected from **VDD** to each pin. DMA cannot interrupt an external memory cycle. DMA does not interfere with internal memory cycles in any way, although a program running in internal memory would have to wait for the end of DMA before accessing external memory. DMA cannot access internal memory.

DMA allows a bootstrap program to be loaded into external RAM ready for execution after reset. If **MemReq** is held high throughout reset, **MemGranted** will be asserted before the bootstrap sequence begins. **MemReq** must be high at least one period **TDCLDCL** of **ClockIn** before **Reset**. The circuit should be designed to ensure correct operation if **Reset** could interrupt a normal DMA cycle.

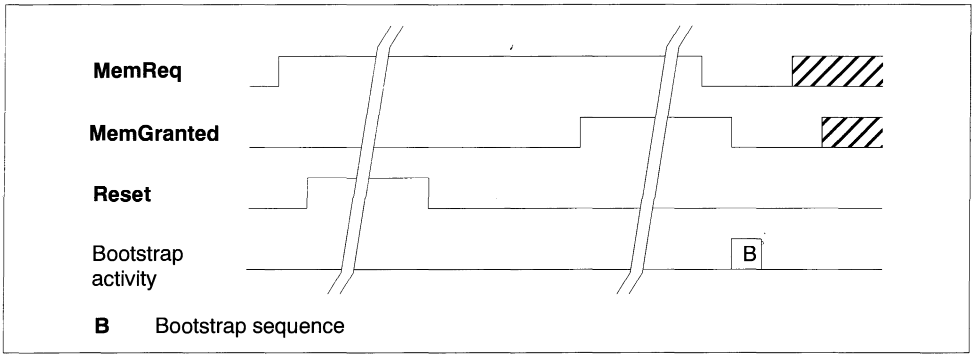


Figure 5.12 IMS T225 DMA sequence at reset

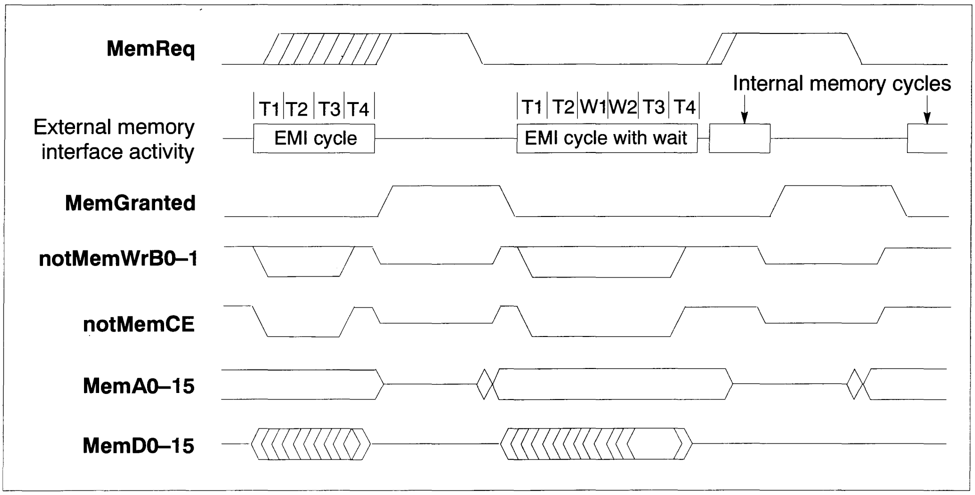


Figure 5.13 IMS T225 operation of **MemReq** and **MemGranted** with external and internal memory cycles

Symbol	Parameter	T225-20		T225–25		T225–30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TMRHMGH	Memory request response time	75	**a**	60	**a**	50	**a**	ns	1
TMRLMGL	Memory request end response time	80	155	65	125	55	105	ns	
TAZMGH	Address bus tristate before **MemGranted**	0		0		0		ns	
TAVMGL	Address bus active after **MemGranted** end	0		0		0		ns	
TDZMGH	Data bus tristate before **MemGranted**	0		0		0		ns	
TEZMGH	Chip enable tristate before **MemGranted**	0		0		0		ns	2
TEVMGL	Chip enable active after **MemGranted** end	–6		–6		–6		ns	
TWEZMGH	Write enable tristate before **MemGranted**	0		0		0		ns	2
TWEVMGL	Write enable active after **MemGranted** end	–6		–6		–6		ns	

Notes

1 Maximum response time **a** depends on whether an external memory cycle is in progress and whether byte access is active. Maximum time is (2 processor cycles) + (number of wait state cycles) for word access; in byte access mode this time is doubled.

2 When using DMA, **notMemCE** and **notMemWrB0-1** should be pulled up with a resistor (typically 10K). Capacitance should be limited to a maximum of 50pF.

Table 5.6 Memory request timing

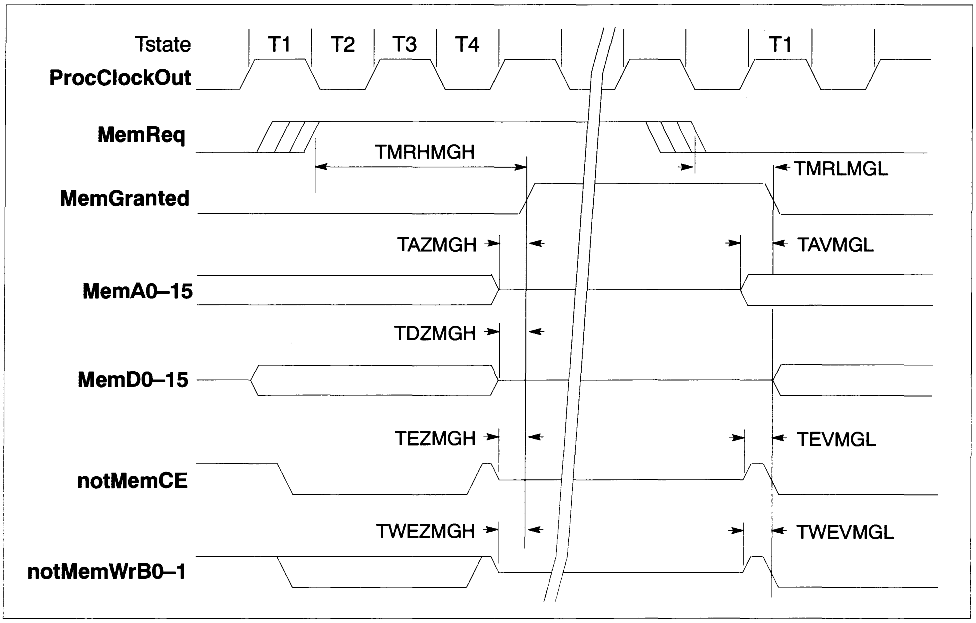


Figure 5.14 IMS T225 memory request timing

6 Events

EventReq and **EventAck** provide an asynchronous handshake interface between an external event and an internal process. When an external event takes **EventReq** high the external event channel (additional to the external link channels) is made ready to communicate with a process. When both the event channel and the process are ready the processor takes **EventAck** high and the process, if waiting, is scheduled. **EventAck** is removed after **EventReq** goes low.

Only one process may use the event channel at any given time. If no process requires an event to occur **EventAck** will never be taken high. Although **EventReq** triggers the channel on a transition from low to high, it must not be removed before **EventAck** is high. **EventReq** should be low during **Reset**; if not it will be ignored until it has gone low and returned high. **EventAck** is taken low when **Reset** occurs.

If the process is a high priority one and no other high priority process is running, the latency is as described on page 70. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

Cycle 1 Sample **EventReq** at pad on the rising edge of **ProcClockOut** and synchronize.

Cycle 2 Edge detect the synchronized **EventReq** and form the interrupt request.

Cycle 3 Sample interrupt vector for microcode ROM in the CPU.

Cycle 4 Execute the interrupt routine for Event rather than the next instruction.

Symbol	Parameter	T225-20		T225-25		T225-30		Units	Notes
		Min	Max	Min	Max	Min	Max		
TVHKH	**EventReq** response	0		0		0		ns	1
TKHVL	**EventReq** hold	0		0		0		ns	1
TVLKL	Delay before removal of **EventAck**	0	157	0	127	0	107	ns	
TKLVH	Delay before re-assertion of **EventReq**	0		0		0		ns	1

Notes

1 Guaranteed, but not tested.

Table 6.1 Event

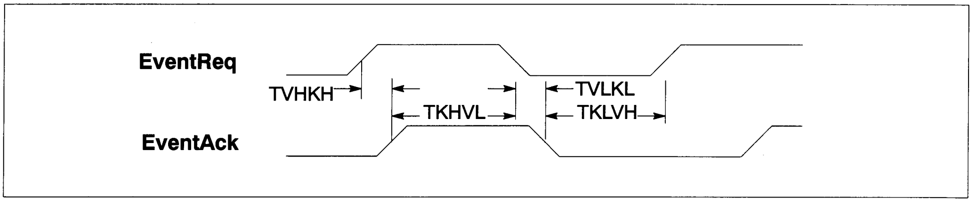


Figure 6.1 IMS T225 event timing

7 Links

Four identical INMOS bi-directional serial links provide synchronised communication between processors and with the outside world. Each link comprises an input channel and output channel. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

The IMS T225 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 5 or 20 Mbits/sec for 20 MHz and 25 MHz devices, and 20 Mbits/sec for faster devices. Links are not synchronised with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

Link speeds can be set by **LinkSpecial**, **Link0Special** and **Link123Special**. The link 0 speed can be set independently. Table 7.1 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed and as **Link123Special** for the others. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

Link	Linkn		Kbytes/sec	
Special	Special	Mbits/sec	Uni	Bi
0	0	10	910	1250
0	1	5	450	670
1	0	10	910	1250
1	1	20	1740	2350

Table 7.1 Speed settings for transputer links

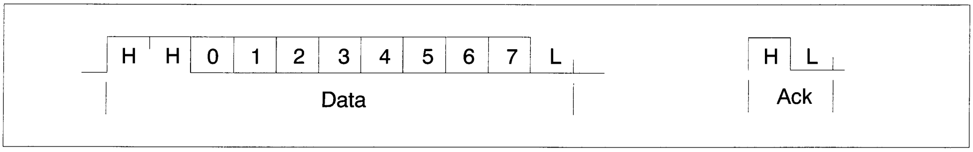


Figure 7.1 IMS T225 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	1
TJDf	**LinkIn** fall time				20	ns	1
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
		5 Mbits/s			30	ns	2
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		ohms	

Notes

1 Guaranteed, but not tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 7.2 Link

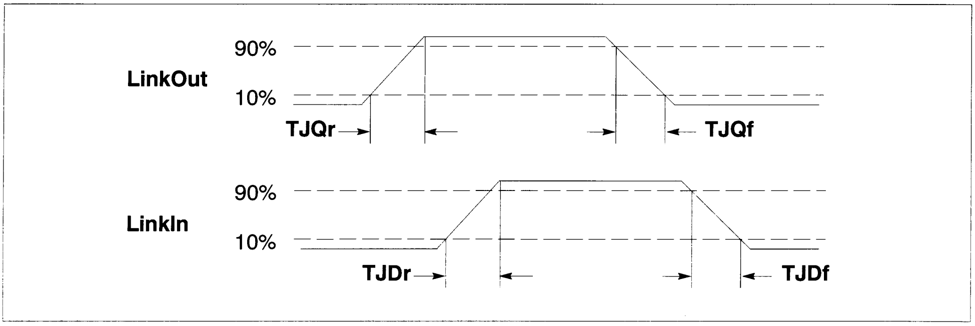


Figure 7.2 IMS T225 link timing

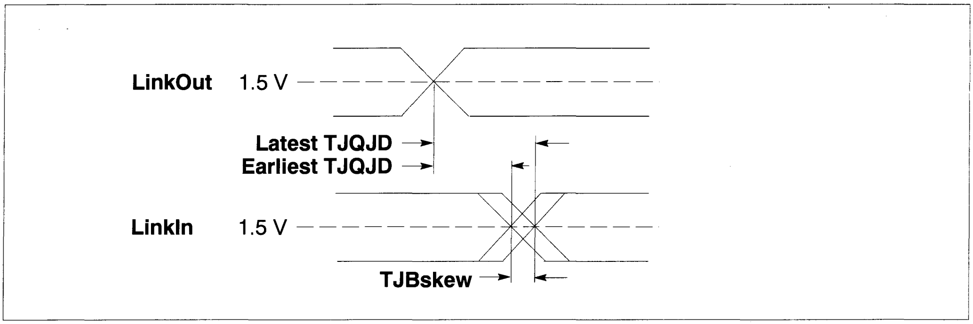


Figure 7.3 IMS T225 buffered link timing

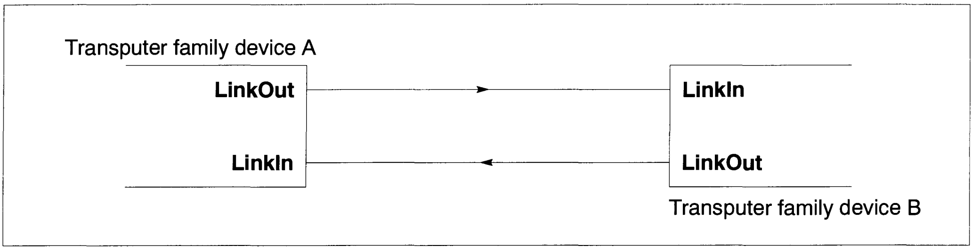


Figure 7.4 IMS T225 links directly connected

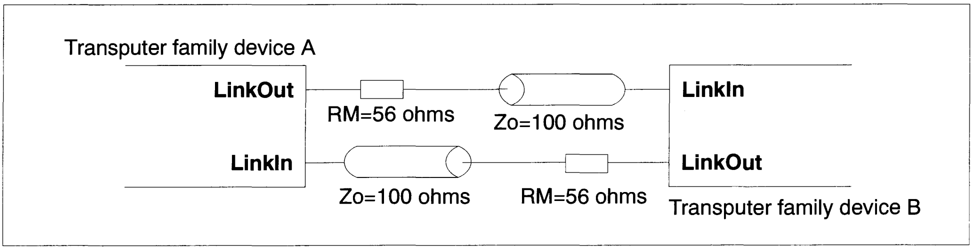


Figure 7.5 IMS T225 links connected by transmission line

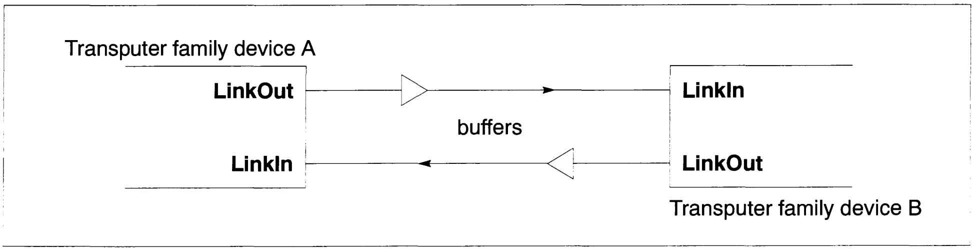


Figure 7.6 IMS T225 links connected by buffers

8 Electrical specifications

8.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1, 2, 3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1, 2, 3
II	Input current		±25	mA	4
OSCT	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 8.1 Absolute maximum ratings

8.2 Operating conditions

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1, 2
CL	Load capacitance on any pin		60	pF	3
TA	Operating temperature range	0	70	°C	4

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Excluding **LinkOut** load capacitance.

4 Air flow rate 400 linear ft/min transverse air flow.

Table 8.2 Operating conditions

8.3 DC electrical characteristics

SYMBOL	PARAMETER		MIN	MAX	UNITS	NOTES
VIH	High level input voltage		2.0	VDD+0.5	V	1, 2
VIL	Low level input voltage		–0.5	0.8	V	1, 2
II	Input current	@ GND<VI<VDD		±10	µA	1, 2
VOH	Output high voltage	@ IOH=2mA	VDD–1		V	1, 2
VOL	Output low voltage	@ IOL=4mA		0.4	V	1, 2
IOZ	Tristate output current	@ GND<V0<VDD		±10	µA	1, 2
PD	Power dissipation			700	mW	2, 3
CIN	Input capacitance	@ f=1MHz		7	pF	4
COZ	Output capacitance	@ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

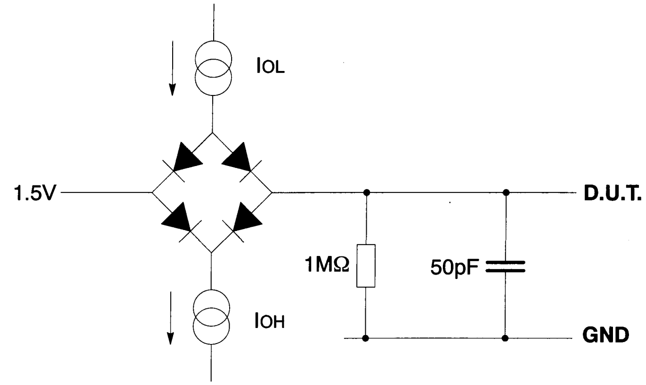
2 Parameters for IMS T225-S measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C. Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution.

4 This parameter is sampled and not 100% tested.

Table 8.3 DC characteristics

8.4 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 8.1 Load circuit for AC measurements

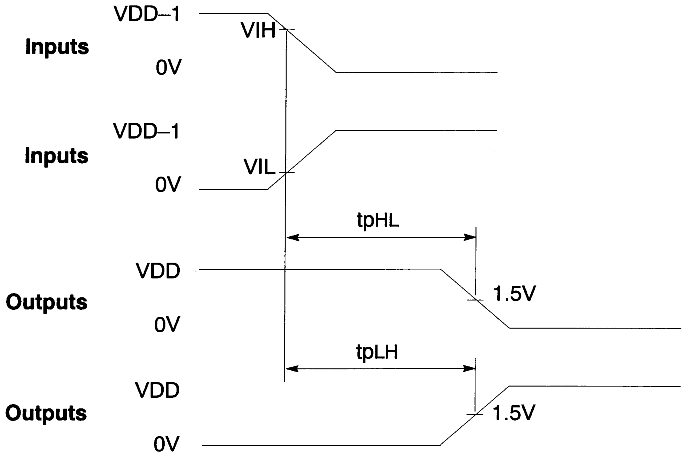


Figure 8.2 AC measurements timing waveforms

8.5 AC timing characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3
TQr	Output rising edges		25	ns	1, 4
TQf	Output falling edges		15	ns	1, 4

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 Guaranteed, but not tested.

4 This parameter is sampled and not 100% tested.

Table 8.4 Input and output edges

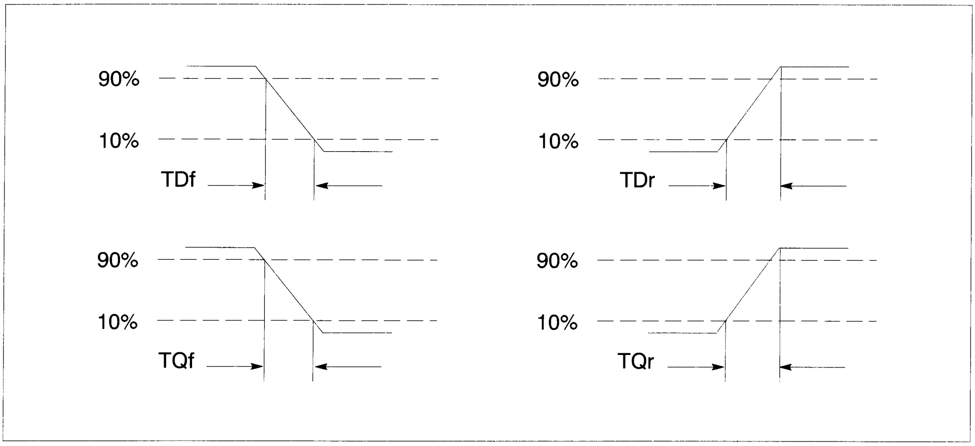


Figure 8.3 IMS T225 input and output edge timing

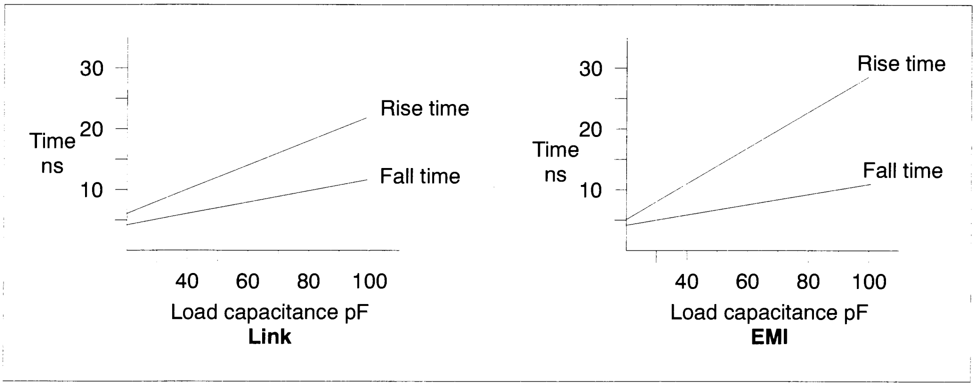


Figure 8.4 Typical rise/fall times

8.6 Power rating

Internal power dissipation P_{INT} of transputer and peripheral chips depends on **VDD**, as shown in figure 8.5. P_{INT} is substantially independent of temperature.

Total power dissipation P_D of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

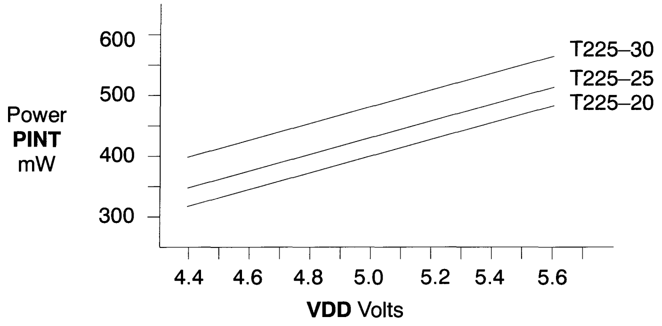
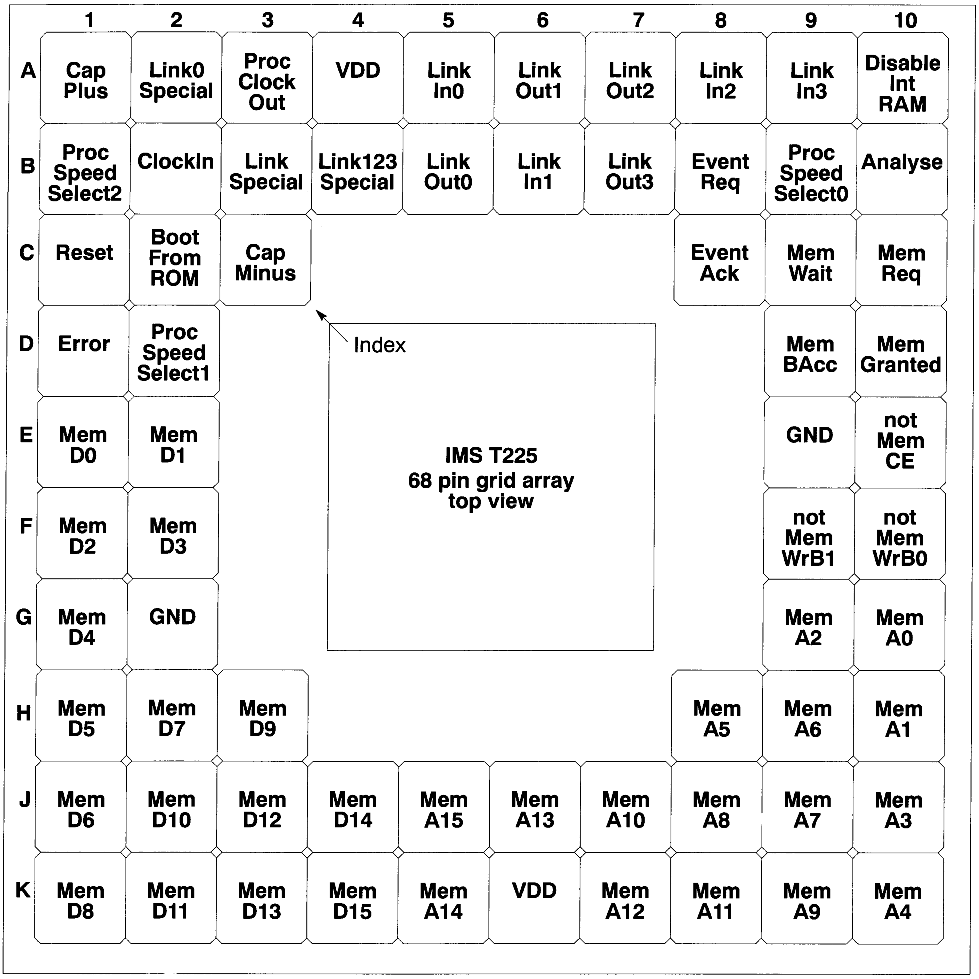


Figure 8.5 IMS T225 internal power dissipation vs VDD

9 Package pinouts

9.1 68 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Cap Plus	Link0 Special	Proc Clock Out	VDD	Link In0	Link Out1	Link Out2	Link In2	Link In3	Disable Int RAM
B	Proc Speed Select2	ClockIn	Link Special	Link123 Special	Link Out0	Link In1	Link Out3	Event Req	Proc Speed Select0	Analyse
C	Reset	Boot From ROM	Cap Minus					Event Ack	Mem Wait	Mem Req
D	Error	Proc Speed Select1							Mem BAcc	Mem Granted
E	Mem D0	Mem D1							GND	not Mem CE
F	Mem D2	Mem D3							not Mem WrB1	not Mem WrB0
G	Mem D4	GND							Mem A2	Mem A0
H	Mem D5	Mem D7	Mem D9					Mem A5	Mem A6	Mem A1
J	Mem D6	Mem D10	Mem D12	Mem D14	Mem A15	Mem A13	Mem A10	Mem A8	Mem A7	Mem A3
K	Mem D8	Mem D11	Mem D13	Mem D15	Mem A14	VDD	Mem A12	Mem A11	Mem A9	Mem A4

Figure 9.1 IMS T225 68 pin grid array package pinout

9.2 68 pin PLCC J-bend package

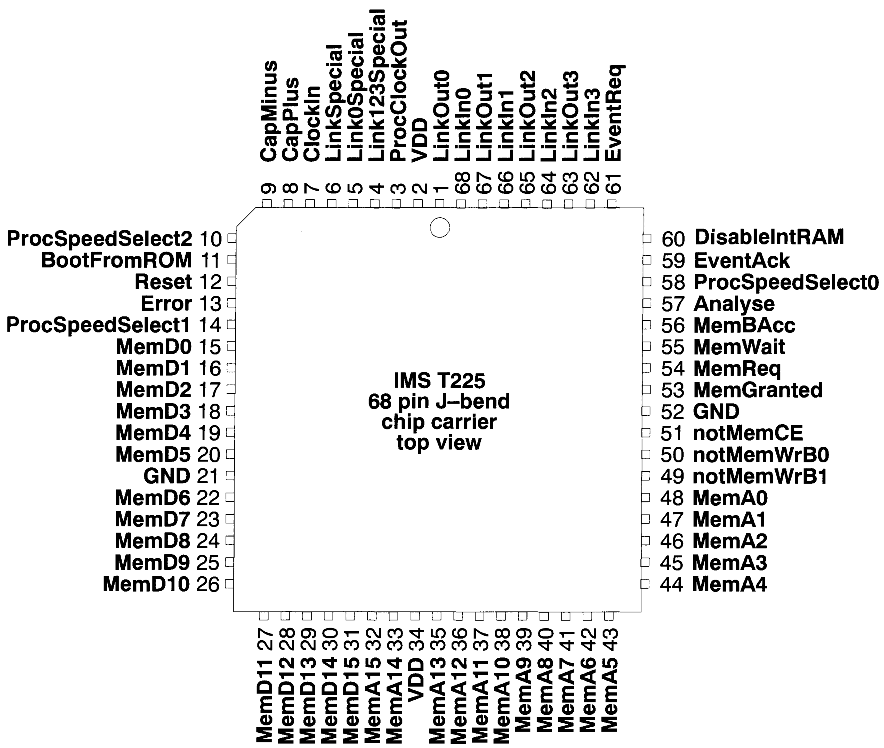
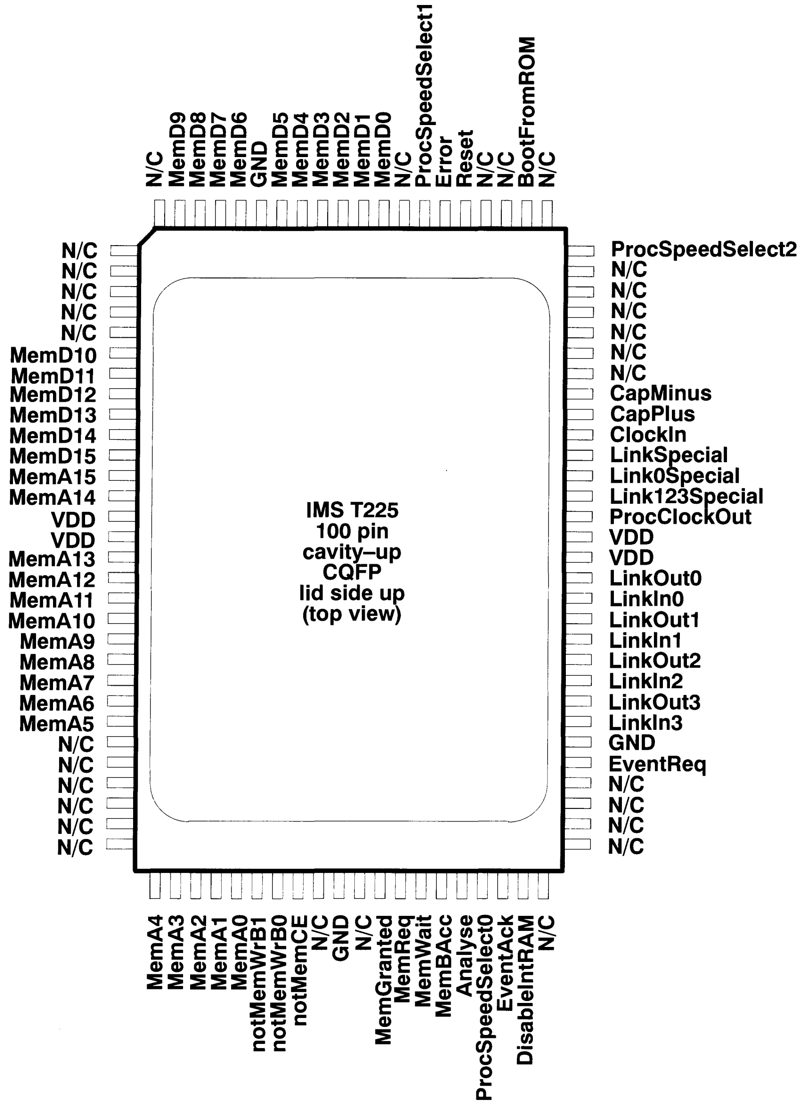


Figure 9.2 IMS T225 68 pin PLCC J-bend package pinout

9.3 100 pin cavity-up ceramic quad flat pack (CQFP) package



N/C indicates pins not connected

Figure 9.3 IMS T225 100 pin cavity-up ceramic quad flat pack package pinout

10 Ordering

This section indicates the designation of speed and package selections for the various devices. Speed of **ClockIn** is 5 MHz for all parts. Transputer processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in the external memory interface chapter 5.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	**Processor clock speed**	**Processor cycle time**	**PLLx**	**Package**
IMS T225-G20S	20.0 MHz	50ns	4.0	68 pin ceramic pin grid array
IMS T225-G25S	25.0 MHz	40ns	5.0	68 pin ceramic pin grid array
IMS T225-G30S	30.0 MHz	33ns	6.0	68 pin ceramic pin grid array
IMS T225-J20S	20.0 MHz	50ns	4.0	68 pin PLCC J–bend
IMS T225-J25S	25.0 MHz	40ns	5.0	68 pin PLCC J–bend
IMS T225-F20S	20.0 MHz	50ns	4.0	100 pin ceramic quad flat pack
IMS T225-F25S	25.0 MHz	40ns	5.0	100 pin ceramic quad flat pack
IMS T225-F30S	30.0 MHz	33ns	6.0	100 pin ceramic quad flat pack

Table 10.1 IMS T225 ordering details

Military versions to MIL-STD-883 are available, see *'The Military and Space Transputer Databook'* for details.

IMS T222 transputer

The IMS T225 is recommended for new designs

inmos®

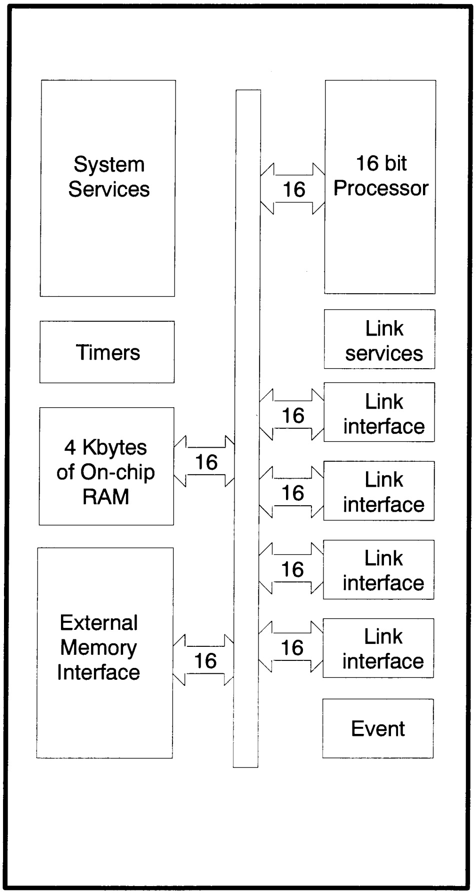
Engineering Data

FEATURES

16 bit architecture
50 ns internal cycle time
20 MIPS peak instruction rate
IMS T220-20 is pin compatible with IMS T225-20
4 Kbytes on-chip static RAM
40 Mbytes/sec sustained data rate to internal memory
64 Kbytes directly addressable external memory
20 Mbytes/sec sustained data rate to external memory
950 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
Bi-directional data rate of 2.4 Mbytes/sec per link
Internal timers of 1μs and 64μs
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ± 5% power supply
Packaging 68 pin PGA / 68 pin PLCC
MIL-STD-883 processing is available

APPLICATIONS

Real time processing
Microprocessor applications
High speed multi processor systems
Industrial control
Robotics
System simulation
Digital signal processing
Telecommunications
Fault tolerant systems
Medical instrumentation



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992

1 Introduction

The IMS T222 transputer is a 16 bit CMOS microcomputer with 4 Kbytes on-chip RAM for high speed processing, an external memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages and provides direct support for the occam model of concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond. A device running at 20 MHz achieves an instruction throughput of 20 MIPS peak. The extended temperature version of the device complies with MIL-STD-883C.

For convenience of description, the IMS T222 operation is split into the basic blocks shown in figure 1.1.

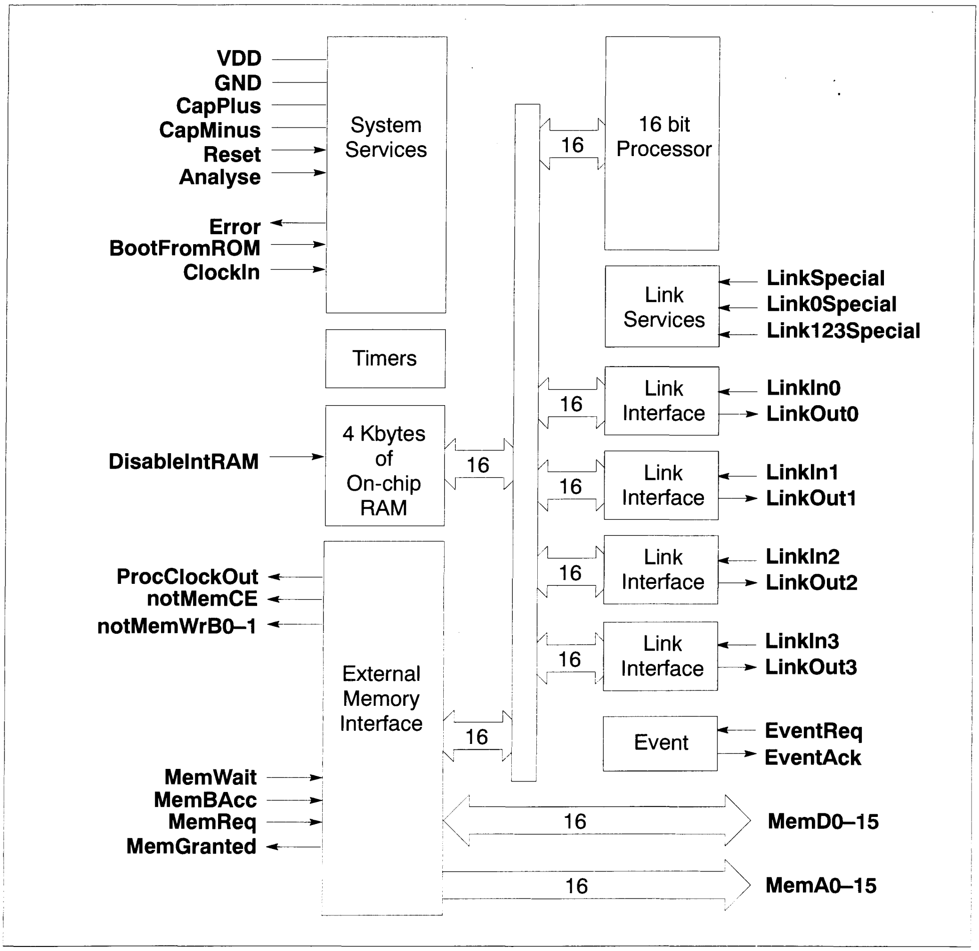


Figure 1.1 IMS T222 block diagram

The IMS T222 can directly access a linear address space of 64 Kbytes. The 16 bit wide non-multiplexed external memory interface provides a data rate of up to 2 bytes every 100 nanoseconds (20 Mbytes/sec) for a 20 MHz device.

System Services include processor reset and bootstrap control, together with facilities for error analysis.

The INMOS communication links allow networks of transputers to be constructed by direct point to point connections with no external logic. The links support the standard operating speed of 10 Mbits/sec, but also operate at 5 or 20 Mbits/sec. The links support overlapped acknowledge; each IMS T222 link can transfer data bi-directionally at up to 2.05 Mbytes/sec.

The IMS T222 is designed to implement the occam language, detailed in the *occam Reference Manual*, but also efficiently supports other languages such as C and Pascal. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*.

This data sheet supplies hardware implementation and characterisation details for the IMS T222. It is intended to be read in conjunction with the Transputer Architecture chapter, which details the architecture of the transputer and gives an overview of occam.

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high.
Pinout details for various packages are given on page 393.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
Reset	in	System reset
Error	out	Error indicator
Analyse	in	Error analysis
BootFromROM	in	Boot from external ROM or from link
DisableIntRAM	in	Disable internal RAM
HoldToGND		Must be connected to **GND**

Table 2.1 IMS T222 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemA0–15	out	Sixteen address lines
MemD0–15	in/out	Sixteen data lines
notMemWrB0–1	out	Two byte-addressing write strobes
notMemCE	out	Chip enable
MemBAcc	in	Byte access mode selector
MemWait	in	Memory cycle extender
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted

Table 2.2 IMS T222 external memory interface

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge

Table 2.3 IMS T222 event

Pin	In/Out	Function
LinkIn0–3	in	Four serial data input channels
LinkOut0–3	out	Four serial data output channels
LinkSpecial	in	Select non-standard speed as 5 or 20 Mbits/sec
Link0Special	in	Select special speed for Link 0
Link123Special	in	Select special speed for Links 1, 2, 3

Table 2.4 IMS T222 link

3 System services

System services include all the necessary logic to initialise and sustain operation of the device. They also include error handling and analysis facilities.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimise inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 20 MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

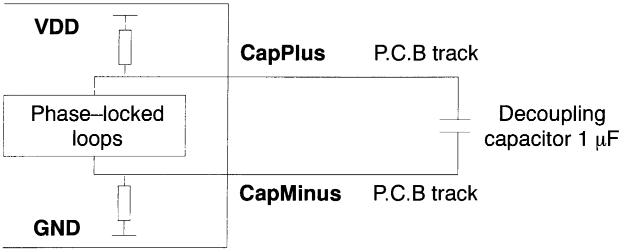


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

		T222-17, -20				
Symbol	**Parameter**	**Min**	**Nom**	**Max**	**Units**	**Notes**
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1,2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1,3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1,4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 8.3).

Table 3.1 Input clock

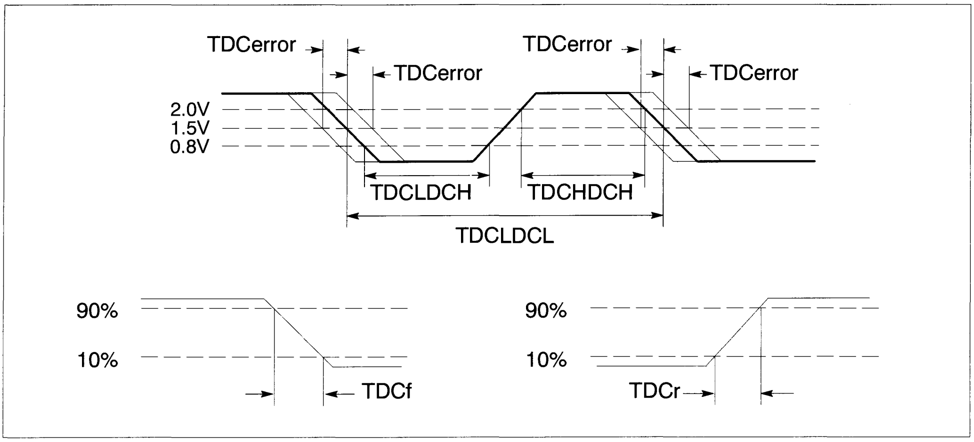


Figure 3.2 **ClockIn** timing

3.4 Bootstrap

The transputer can be bootstrapped either from a link or from external ROM. To facilitate debugging, **BootFromROM** may be dynamically changed but must obey the specified timing restrictions. It is sampled once only by the transputer, before the first instruction is executed after **Reset** is taken low.

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the **W** register points to *MemStart* (page 370).

If **BootFromROM** is connected low (e.g. to **GND**) the transputer will wait for the first bootstrap message to arrive on any one of its links. The transputer is ready to receive the first byte on a link within two processor cycles **TPCLPCL** after **Reset** goes low.

If the first byte received (the control byte) is greater than 1 it is taken as the quantity of bytes to be input. The following bytes, to that quantity, are then placed in internal memory starting at location *MemStart*. Following reception of the last byte the transputer will start executing code at *MemStart* as a low priority process. **BootFromROM** may be taken high after reception of the last byte, if required. The memory space immediately above the loaded code is used as work space. A byte arriving on other links after the control byte has been received and on the bootstrapping link after the last bootstrap byte, will be retained and no acknowledge will be sent until a process inputs from them.

3.5 Peek and poke

Any location in internal or external memory can be interrogated and altered when the transputer is waiting for a bootstrap from link. If the control byte is 0 then four more bytes are expected on the same link. The first two byte word is taken as an internal or external memory address at which to poke (write) the second two byte word. If the control byte is 1 the next two bytes are used as the address from which to peek (read) a word of data; the word is sent down the output channel of the same link.

Following such a peek or poke, the transputer returns to its previously held state. Any number of accesses may be made in this way until the control byte is greater than 1, when the transputer will commence reading its bootstrap program. Any link can be used, but addresses and data must be transmitted via the same link as the control byte.

3.6 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initialises the transputer and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) must not occur whilst **Reset** is high but can occur before bootstrap (page 382). If **BootFromROM** is high bootstrapping will take place immediately after **Reset** goes low, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

3.7 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 48). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags **Error** and **HaltOnError** are not altered at reset, whether **Analyse** is asserted or not.

Input links will continue with outstanding transfers. Output links will not make another access to memory for data but will transmit only those bytes already in the link buffer. Providing there is no delay in link acknowledgement, the links should be inactive within a few microseconds of the transputer halting.

Reset should not be asserted before the transputer has halted and link transfers have ceased. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 3.2.

I	**MemStart** if bootstrapping from a link, or the external memory bootstrap address if bootstrapping from ROM.
W	**MemStart** if bootstrapping from ROM, or the address of the first free word after the bootstrap program if bootstrapping from link.
A	The value of **I** when the processor halted.
B	The value of **W** when the processor halted, together with the priority of the process when the transputer was halted (i.e. the **W** descriptor).
C	The ID of the bootstrapping link if bootstrapping from link.

Table 3.2 Register values after **Analyse**

Symbol	Parameter	T222-17, -20			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

1 Full periods of **ClockIn TDCLDCL** required.

2 At power-on reset.

3 Must be stable until after end of bootstrap period. See Bootstrap section 3.4.

Table 3.3 **Reset** and **Analyse**

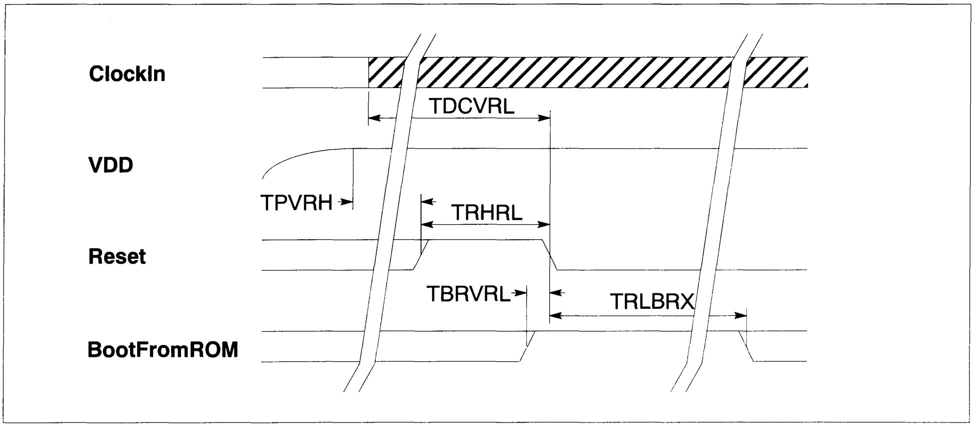


Figure 3.3 Transputer **Reset** timing with **Analyse** low

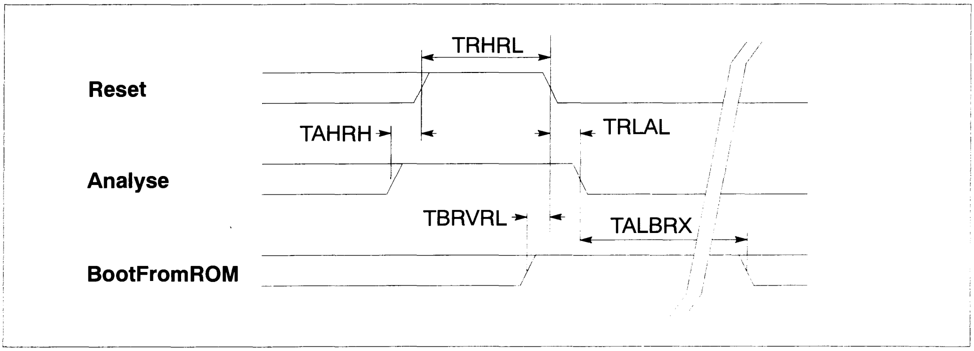


Figure 3.4 Transputer **Reset**, **Analyse** and **BootFromROM** timing

3.8 Error

The **Error** pin is connected directly to the internal *Error* flag and follows the state of that flag. If **Error** is high it indicates an error in one of the processes caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 48). Once set, the *Error* flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (page 367).

A process can be programmed to stop if the *Error* flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data.

By setting the *HaltOnError* flag the transputer itself can be programmed to halt if *Error* becomes set. If *Error* becomes set after *HaltOnError* has been set, all processes on that transputer will cease but will not necessarily cause other transputers in a network to halt. Setting *HaltOnError* after *Error* will not cause the transputer to halt; this allows the processor reset and analyse facilities to function with the flags in indeterminate states.

An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by applying the **Error** output signal of the errant transputer to the **EventReq** pin of a suitably programmed master transputer. Since the process state is preserved when stopped by an error, the master transputer can then use the analyse function to debug the fault. When using such a circuit, note that the *Error* flag is in an indeterminate state on power up; the circuit and software should be designed with this in mind.

Error checks can be removed completely to optimise the performance of a proven program; any unexpected error then occurring will have an arbitrary undefined effect.

If a high priority process pre-empts a low priority one, status of the *Error* and *HaltOnError* flags is saved for the duration of the high priority process and restored at the conclusion of it. Status of the *Error* flag is transmitted to the high priority process but the *HaltOnError* flag is cleared before the process starts. Either flag can be altered in the process without upsetting the error status of any complex operation being carried out by the pre-empted low priority process.

In the event of a transputer halting because of *HaltOnError*, the links will finish outstanding transfers before shutting down. If **Analyse** is asserted then all inputs continue but outputs will not make another access to memory for data.

After halting due to the *Error* flag changing from 0 to 1 whilst *HaltOnError* is set, register **I** points two bytes past the instruction which set *Error*. After halting due to the **Analyse** pin being taken high, register **I** points one byte past the instruction being executed. In both cases **I** will be copied to register **A**.

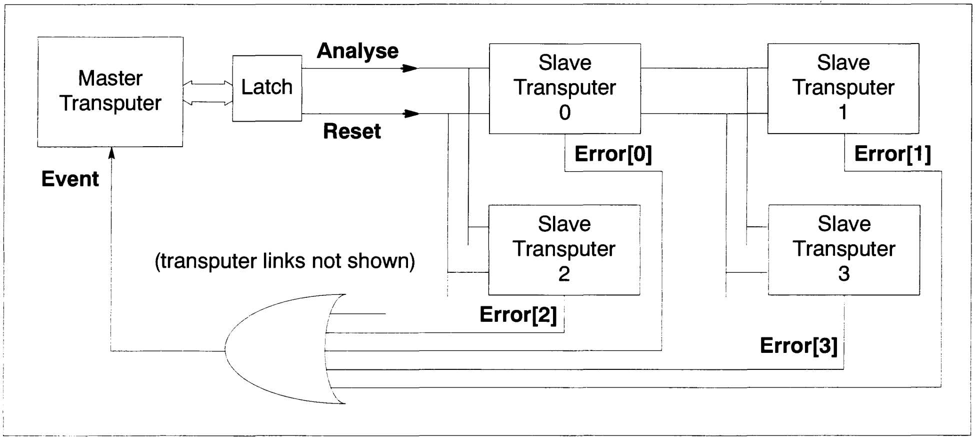


Figure 3.5 Error handling in a multi-transputer system

4 Memory

The IMS T222 has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (page 372). The transputer can also access an additional 60 Kbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

IMS T222 memory is byte addressed, with words aligned on two-byte boundaries. The least significant byte of a word is the lowest addressed byte.

The bits in a byte are numbered 0 to 7, with bit 0 the least significant. The bytes are numbered from 0, with byte 0 the least significant. In general, wherever a value is treated as a number of component values, the components are numbered in order of increasing numerical significance, with the least significant component numbered 0. Where values are stored in memory, the least significant component value is stored at the lowest (most negative) address.

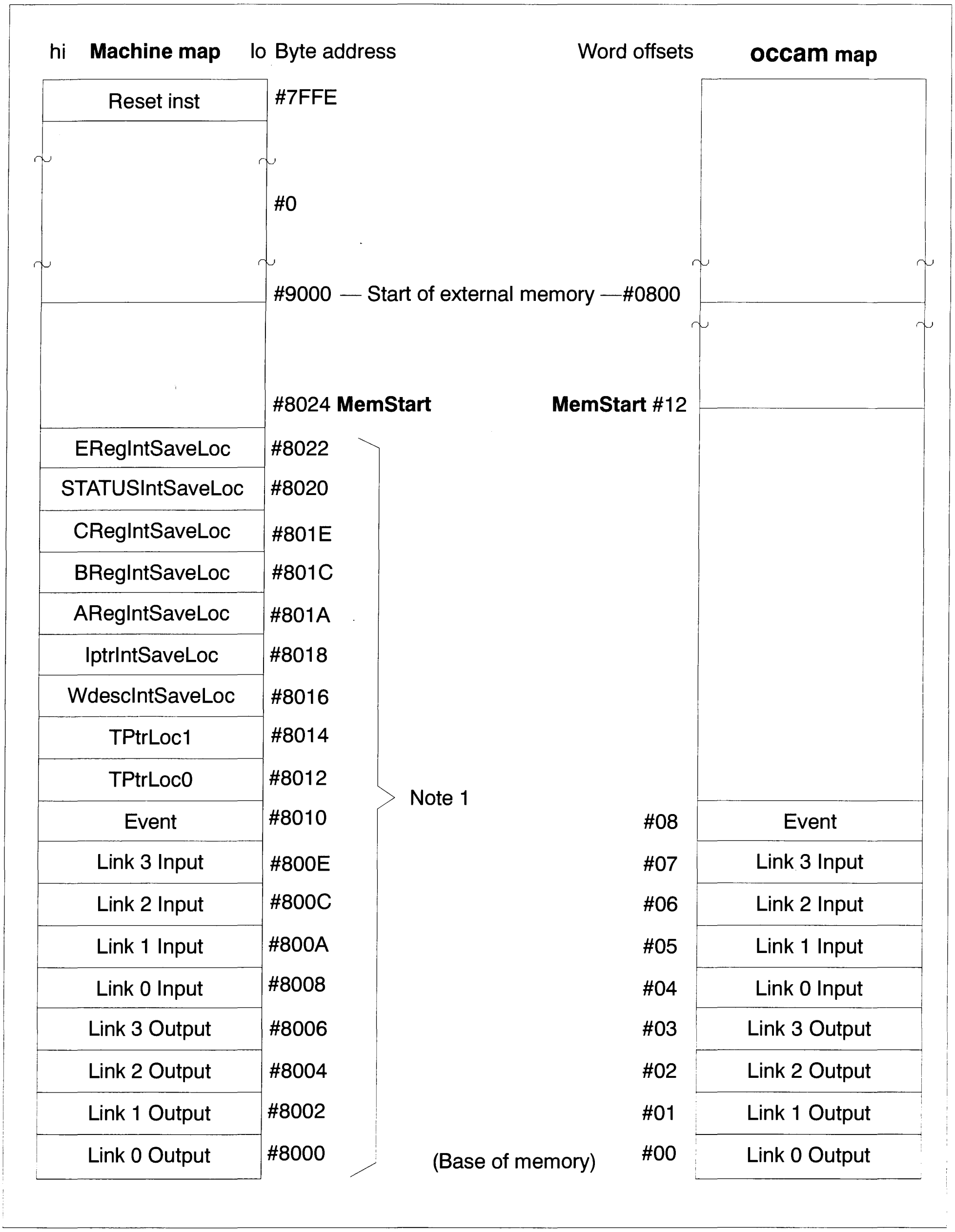
Internal memory starts at the most negative address #8000 and extends to #8FFF. User memory begins at #8024; this location is given the name *MemStart*.

The reserved area of internal memory below *MemStart* is used to implement link and event channels.

Two words of memory are reserved for timer use, *TPtrLoc0* for high priority processes and *TPtrLoc1* for low priority processes. They either indicate the relevant priority timer is not in use or point to the first process on the timer queue at that priority level.

Values of certain processor registers for the current low priority process are saved in the reserved *IntSaveLoc* locations when a high priority process pre-empts a low priority one.

External memory space starts at #9000 and extends up through #0000 to #7FFF. ROM bootstrapping code must be in the most positive address space, starting at #7FFE. Address space immediately below this is conventionally used for ROM based code.


hi Machine map lo Byte address
Word offsets occam map
Reset inst
#7FFE
#0
#9000 — Start of external memory —#0800
#8024 MemStart
MemStart #12
ERegIntSaveLoc #8022
STATUSIntSaveLoc #8020
CRegIntSaveLoc #801E
BRegIntSaveLoc #801C
ARegIntSaveLoc #801A
IptrIntSaveLoc #8018
WdescIntSaveLoc #8016
TPtrLoc1 #8014
TPtrLoc0 #8012
Note 1
Event #8010 #08 Event
Link 3 Input #800E #07 Link 3 Input
Link 2 Input #800C #06 Link 2 Input
Link 1 Input #800A #05 Link 1 Input
Link 0 Input #8008 #04 Link 0 Input
Link 3 Output #8006 #03 Link 3 Output
Link 2 Output #8004 #02 Link 2 Output
Link 1 Output #8002 #01 Link 1 Output
Link 0 Output #8000 (Base of memory) #00 Link 0 Output

Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 367). For details see the *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 4.1 IMS T222 memory map

5 External memory interface

The IMS T222 External Memory Interface (EMI) allows access to a 16 bit address space via separate address and data buses. The data bus can be configured for either 16 bit or 8 bit memory access, allowing the use of a single bank of byte-wide memory. Both word-wide and byte-wide access may be mixed in a single memory system (page 378).

5.1 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn**. Its period is equal to one internal microcode cycle time, and can be derived from the formula

TPCLPCL = TDCLDCL / PLLx

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (Ordering section).

Edges of the various external memory strobes are synchronized by, but do not all coincide with, rising or falling edges of **ProcClockOut**.

		T222-17		**T222-20**			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TPCLPCL	**ProcClockOut** period	55	59	48	52	ns	2
TPCHPCL	**ProcClockOut** pulse width high	21	36	18	32	ns	2
TPCLPCH	**ProcClockOut** pulse width low	**a**		**a**		ns	2, 3, 4
TPCstab	**ProcClockOut** stability		8		8	%	1, 2

Notes

1 Stability is the variation of cycle periods between two consecutive cycles, measured at corresponding points on the cycles.

2 This parameter is sampled and not 100% tested.

3 **a** is TPCLPCL – TPCHPCL.

4 This is a nominal value.

Table 5.1 **ProcClockOut**

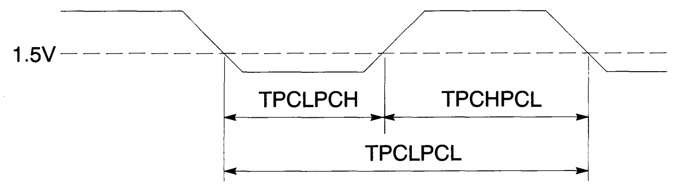


Figure 5.1 IMS T222 **ProcClockOut** timing

5.2 Tstates

The external memory cycle is divided into four **Tstates** with the following functions:

T1	Address and control setup time.
T2	Data setup time.
T3	Data read/write.
T4	Data and address hold after access.

Each **Tstate** is half a processor cycle **TPCLPCL** long. An external memory cycle is always a complete number of cycles **TPCLPCL** in length and the start of **T1** always coincides with a rising edge of **ProcClockOut**. **T2** can be extended indefinitely by adding externally generated wait states of one complete processor cycle each.

5.3 Internal access

During an internal memory access cycle the external memory interface address bus **MemA0-15** reflects the word address used to access internal RAM, **notMemWrB0-1** and **notMemCE** are inactive and the data bus **MemD0-15** is tristated. This is true unless and until a DMA (memory request) activity takes place, when the lines will be placed in a high impedance state by the transputer. Bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 367).

5.4 MemA0–15

External memory addresses are output on a non-multiplexed 16 bit bus. The address is valid at the start of **T1** and remains so until the end of **T4**, with the timing shown. Byte addressing is carried out internally by the IMS T222 for read cycles. For write cycles the relevant bytes in memory are addressed by the write enables **notMemWrB0-1**.

The transputer places the address bus in a high impedance state during DMA.

5.5 MemD0–15

The non-multiplexed data bus is 16 bits wide. Read cycle data may be set up on the bus at any time after the start of **T1**, but must be valid when the IMS T222 reads it during **T4**. Data can be removed any time after the rising edge of **notMemCE**, but must be off the bus no later than the middle of **T1**, which allows for bus turn-around time before the data lines are driven at the start of **T2** in a processor write cycle.

Write data is placed on the bus at the start of **T2** and removed at the end of **T4**. It is normally written into memory in synchronism with **notMemCE** going high.

The data bus is high impedance except when the transputer is writing data. If only one byte is being written, the unused 8 bits of the bus are high impedance at that time. In byte access mode **MemD8-15** are high impedance during the external memory cycle which writes the most significant (second) byte (page 378).

If the data setup time for read or write is too short it can be extended by inserting wait states at the end of **T2** (page 380).

Symbol	Parameter	T222-17		T222-20		Units	Notes
		Min	Max	Min	Max		
TAVEL	Address valid before chip enable low	12		8		ns	1
TELEH	Chip enable low	83	88	68	80	ns	1
TEHEL	Delay before chip enable re-assertion	24		19		ns	1,2
TEHAX	Address hold after chip enable high	12		3		ns	1
TELDrV	Data valid from chip enable low	0	53	0	50	ns	
TAVDrV	Data valid from address valid	0	65	0	63	ns	
TDrVEH	Data setup before chip enable high	30		22		ns	
TEHDrZ	Data hold after chip enable high	0	24	0	20	ns	
TWEHEL	Write enable setup before chip enable low	24		18		ns	3
TPCHEL	ProcClockOut high to chip enable low	12		8	20	ns	1

Notes

1 This parameter is common to read and write cycles and to byte-wide memory accesses.

2 These values assume back-to-back external memory accesses.

3 Timing is for both write enables **notMemWrB0-1**.

Table 5.2 Read

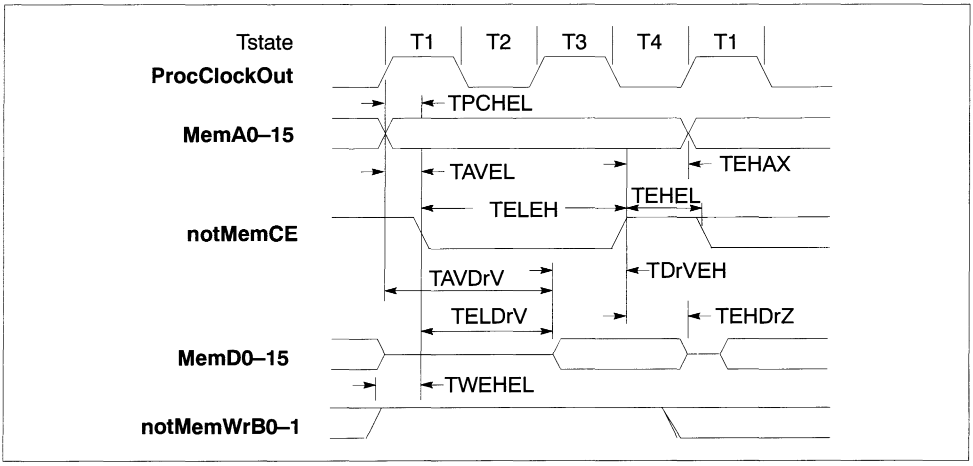


Figure 5.2 IMS T222 external read cycle

5.6 notMemWrB0–1

Two write enables are provided, one to write each byte of the word. When writing a word, both write enables are asserted; when writing a byte only the appropriate write enable is asserted. **notMemWrB0** addresses the least significant byte.

The write enables are synchronized with the chip enable signal **notMemCE**, allowing them to be used without **notMemCE** for simple designs.

Data may be strobed into memory using **notMemWrB0-1** without the use of **notMemCE**, as the write enables go high between consecutive external memory write cycles. The write enables are placed in a high impedance state during DMA, and are inactive during internal memory access.

Symbol	Parameter	T222-17		T222-20		Units	Notes
		Min	Max	Min	Max		
TDwVEH	Data setup before chip enable high	57		50		ns	
TEHDwZ	Data hold after write	12	17	5	25	ns	
TDwZEL	Write data invalid to next chip enable	12		1		ns	
TWELEL	Write enable setup before chip enable low	–4	0	–8	3	ns	1
TEHWEH	Write enable hold after chip enable high	0	4	–3	6	ns	1

Notes

1 Timing is for both write enables **notMemWrB0-1**.

Table 5.3 Write

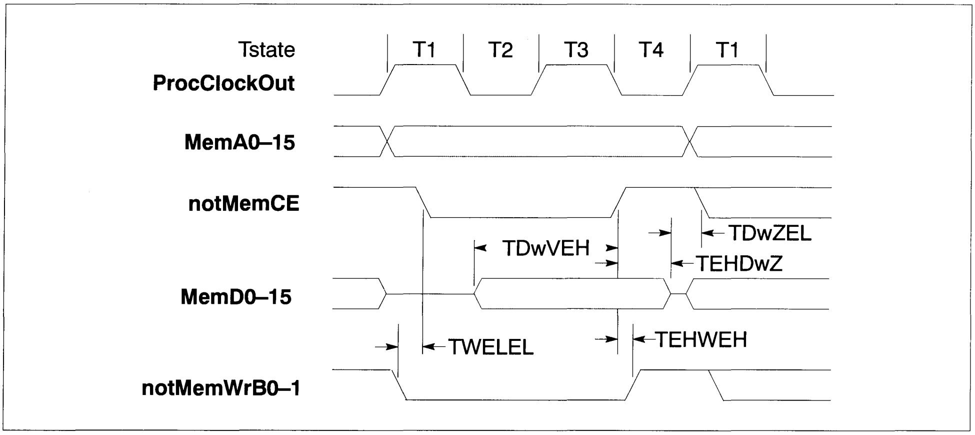


Figure 5.3 IMS T222 external write cycle

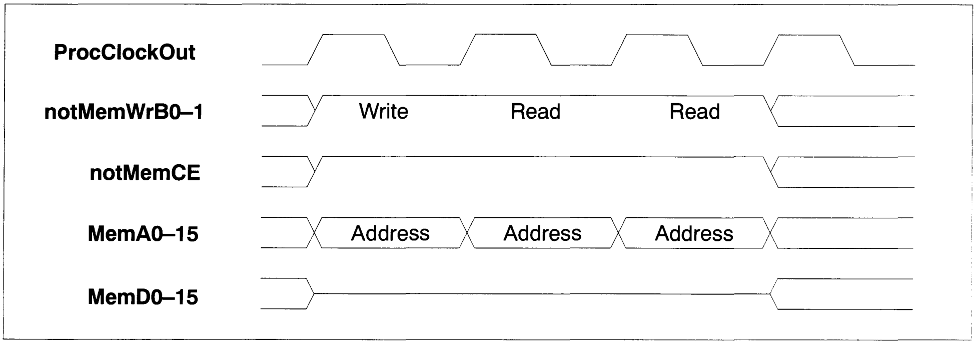


Figure 5.4 IMS T222 bus activity for 3 internal memory cycles

5.7 notMemCE

The active low signal **notMemCE** is used to enable external memory on both read and write cycles.

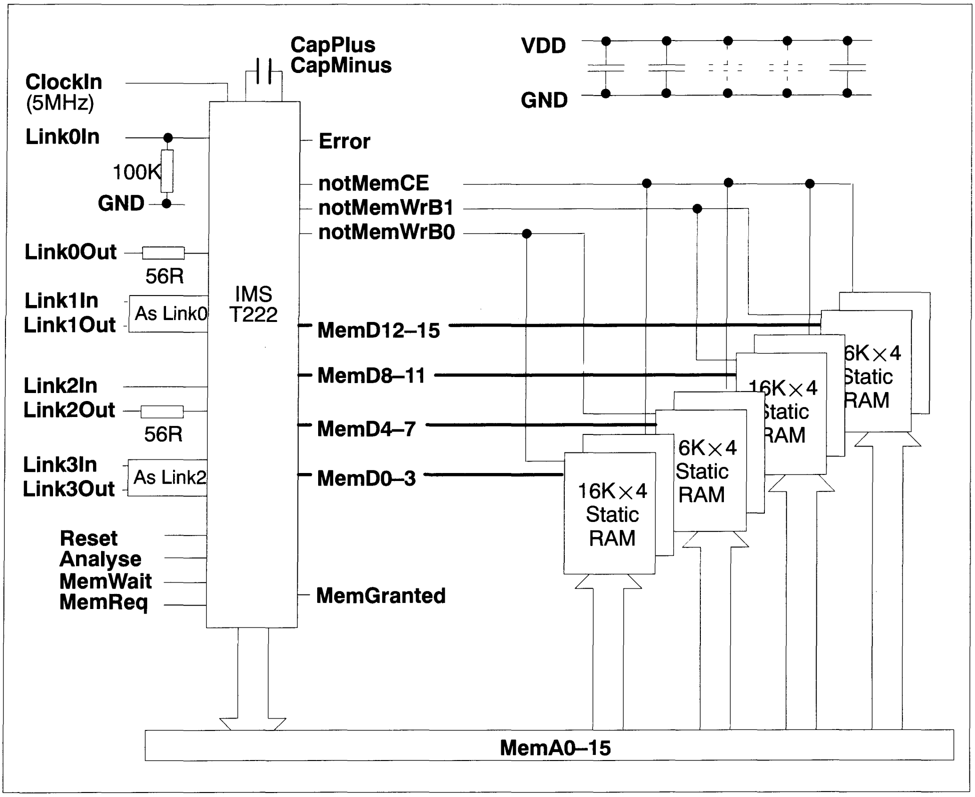


Figure 5.5 IMS T222 static RAM application

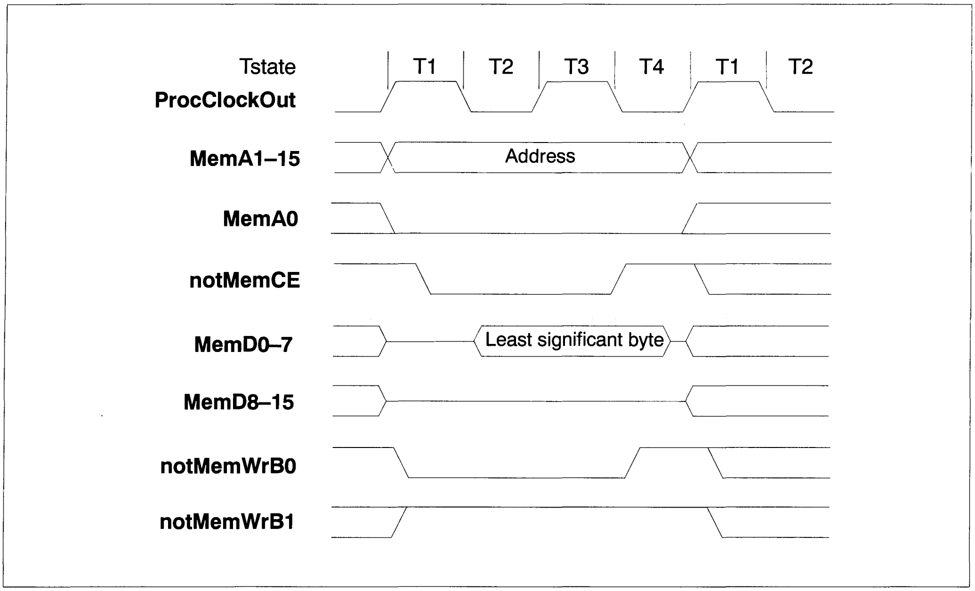


Figure 5.6 IMS T222 Least significant byte write in word access mode

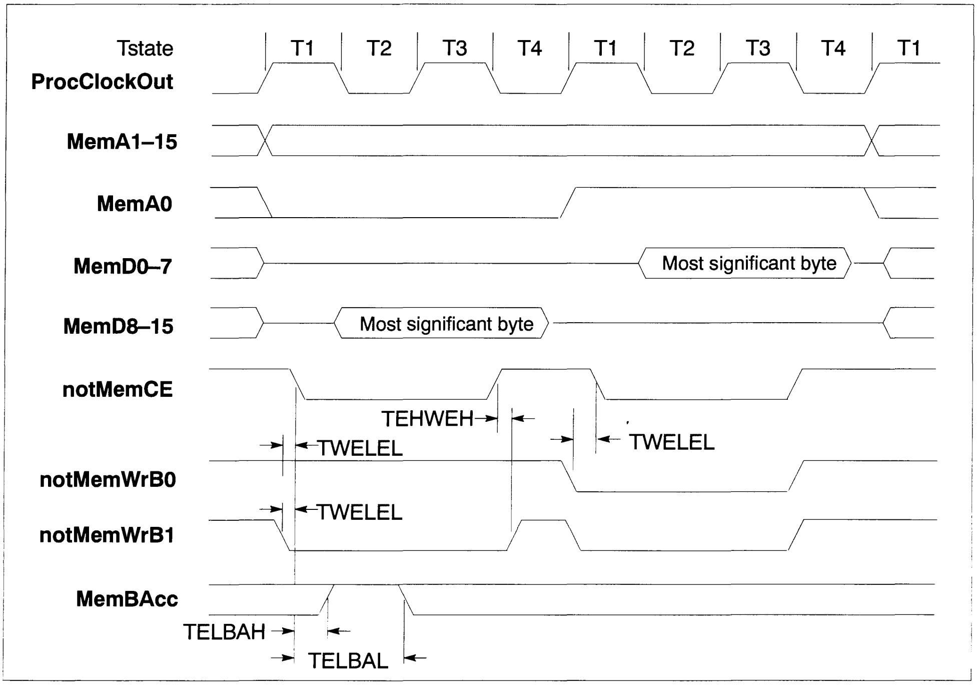


Figure 5.7 IMS T222 Most significant byte write to byte-wide memory

5.8 MemBAcc

The IMS T222 will, by default, perform word access at even memory locations. Access to byte-wide memory can be achieved by taking **MemBAcc** high with the timing shown. Where all external memory operations are to byte-wide memory, **MemBAcc** may be wired permanently high. The state of this signal is latched during **T2**. Where external memory operations may be to both byte and word wide memory, **MemBAcc** should be obtained by address decoding. If you use a memory system in which word wide memory may be used in byte access mode it is recommended that **notMemWrB1** is OR gated with **MemA0**, to prevent any spurious data being written to the RAM.

If **MemBAcc** is low then a full word will be accessed in one external memory cycle, otherwise the high and low bytes of the word will be separately accessed during two consecutive cycles. The first (least significant) byte is accessed at the word address (**MemA0** is low). The second (most significant) byte is accessed at the word address +1 (**MemA0** is high).

5.8.1 Word Read/Write in Byte Access Mode

With **MemBAcc** high, the first cycle is identical with a normal word access cycle. However, it will be immediately followed by another memory cycle, which will use **MemD0-7** to read or write the second (most significant) byte of data. During this second cycle, for a write, **notMemWrB0–1** both go low as in the first cycle and **MemA0** goes high. For a read, **notMemWrB0–1** remain high and **MemA0** goes high. **MemD8–15** are high impedance for both read and write in the second cycle.

5.8.2 Byte Write in Byte Access Mode

Writing a Most Significant Byte

In the first cycle **notMemWrB1** will go low and **notMemWrB0** will remain high. **MemA0** remains low. In the second cycle **MemA0** goes high and **notMemWrB0–1** go low. The data is written on **MemD0–7** in the second cycle.

Writing a Least Significant Byte

In the first cycle **notMemWrB1** remains high and **notMemWrB0** goes low. **MemA0** remains low. In the second cycle **MemA0** and **notMemWrB0** go high, **notMemWrB1** remains high. Data is written on **MemD0–7** in the first cycle.

		T222-17		T222-20			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TELBAH	**MemBAcc** high from chip enable		15		12	ns	
TELBAL	**MemBAcc** low from chip enable	29		32		ns	

Table 5.4 Byte-wide memory access

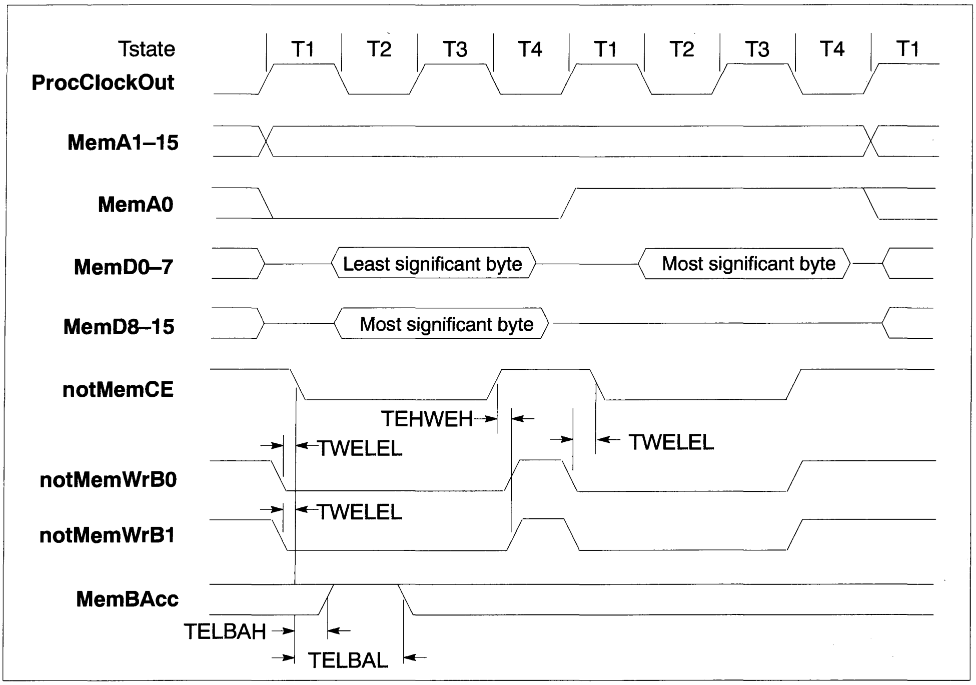


Figure 5.8 IMS T222 word write to byte-wide memory

5.9 MemWait

Taking **MemWait** high with the timing shown in the diagram will extend the duration of **T2** by one processor cycle **TPCLPCL**. One wait state comprises the pair **W1** and **W2**. **MemWait** is sampled during **T2**, and should not change state in this region. If **MemWait** is still high when sampled in **W2** then another wait period will be inserted. This can continue indefinitely. Internal memory access is unaffected by the number of wait states selected. The wait state generator can be a simple digital delay line, synchronized to **notMemCE**. The **Single Wait State Generator** circuit in figure 5.10 can be extended to provide two or more wait states, as shown in figure 5.11.

		T222-17		T222-20			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TPCHWtH	**MemWait** asserted after **ProcClockOut** high		27		25	ns	
TPCHWtL	**MemWait** low after **ProcClockOut** high	39		45		ns	

Table 5.5 Memory wait

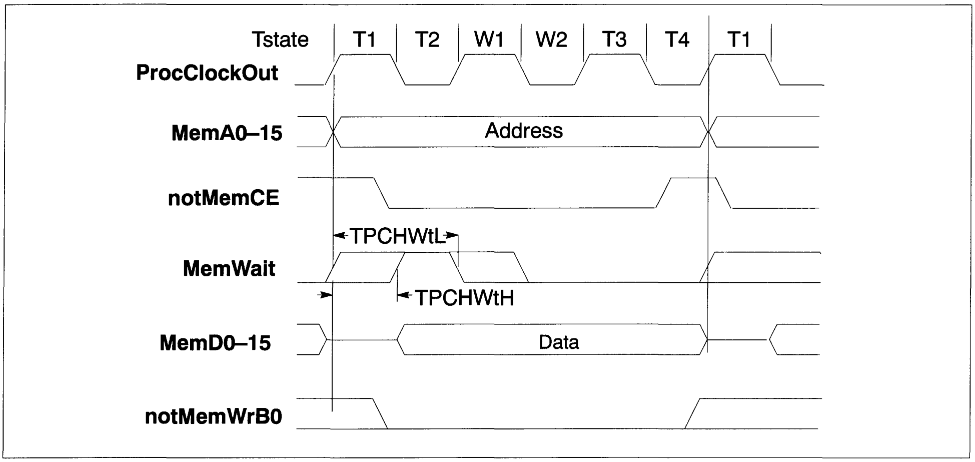


Figure 5.9 IMS T222 memory wait timing

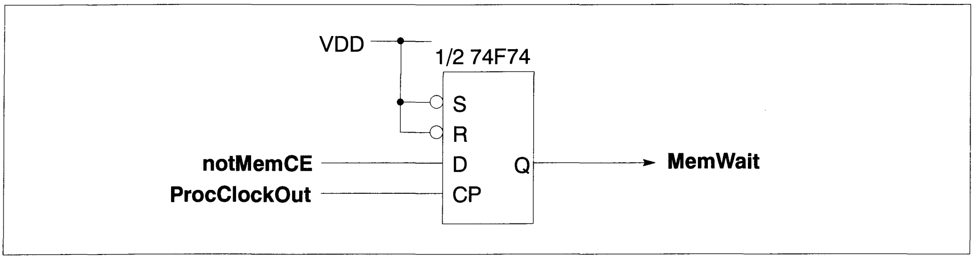


Figure 5.10 Single wait state generator

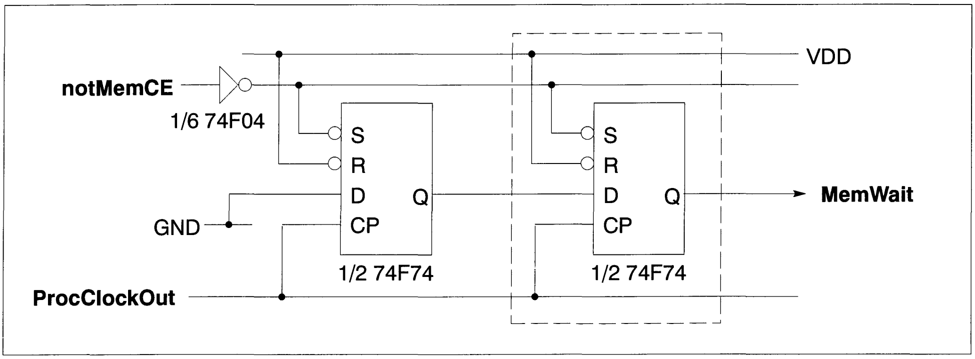


Figure 5.11 Extendable wait state generator

5.10 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. For external memory cycles, the IMS T222 samples **MemReq** during the first high phase of **ProcClockOut** after **notMemCE** goes low. In the absence of an external memory cycle, **MemReq** is sampled during every rising edge of **ProcClockOut**. **MemA0-15**, **MemD0-15**, **notMemWrB0-1** and **notMemCE** are tristated before **MemGranted** is asserted.

Removal of **MemReq** is sampled at each rising edge of **ProcClockOut** and **MemGranted** removed with the timing shown. Further external bus activity, either external cycles or reflection of internal cycles, will commence during the next low phase of **ProcClockOut**.

Chip enable, write enables, address bus and data bus are in a high impedance state during DMA. External circuitry must ensure that **notMemCE** and **notMemWrB0-1** do not become active whilst control is being transferred; it is recommended that a 10K resistor is connected from **VDD** to each pin. DMA cannot interrupt an external memory cycle. DMA does not interfere with internal memory cycles in any way, although a program running in internal memory would have to wait for the end of DMA before accessing external memory. DMA cannot access internal memory.

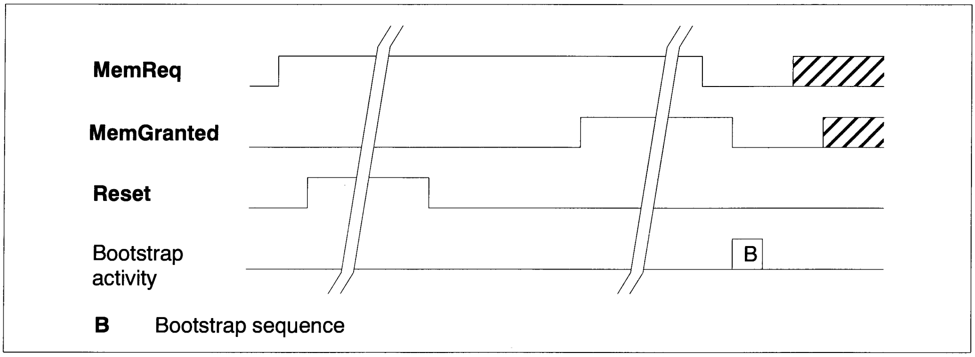


Figure 5.12 IMS T222 DMA sequence at reset

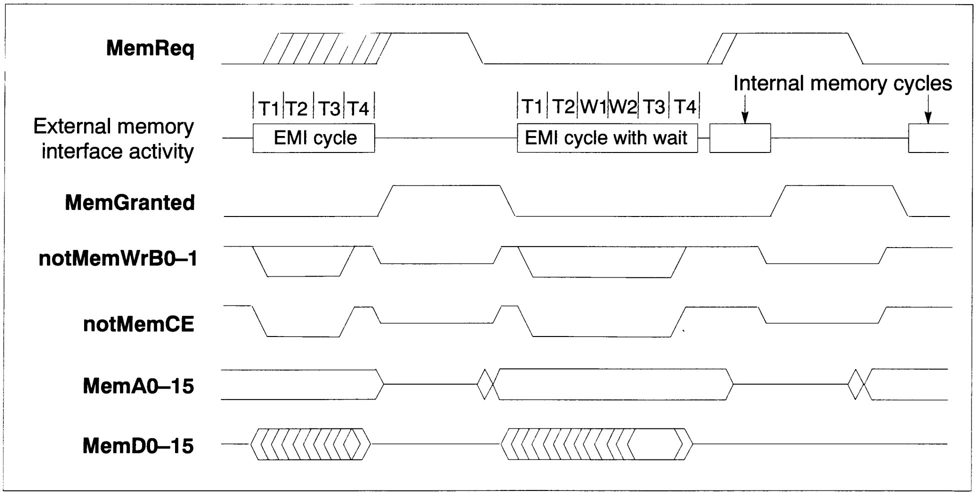


Figure 5.13 IMS T222 operation of **MemReq** and **MemGranted** with external and internal memory cycles

DMA allows a bootstrap program to be loaded into external RAM ready for execution after reset. If **MemReq** is held high throughout reset, **MemGranted** will be asserted before the bootstrap sequence begins. **MemReq** must be high at least one period **TDCLDCL** of **ClockIn** before **Reset**. The circuit should be designed to ensure correct operation if **Reset** could interrupt a normal DMA cycle.

		T222–17		**T222-20**			
Symbol	**Parameter**	**Min**	**Max**	**Min**	**Max**	**Units**	**Notes**
TMRHMGH	Memory request response time	100	**a**	75	**a**	ns	1
TMRLMGL	Memory request end response time	100	114	80	155	ns	
TAZMGH	Address bus tristate before **MemGranted**	0		0		ns	
TAVMGL	Address bus active after **MemGranted** end	0		0		ns	
TDZMGH	Data bus tristate before **MemGranted**	0		0		ns	
TEZMGH	Chip enable tristate before **MemGranted**	0		0		ns	2
TEVMGL	Chip enable active after **MemGranted** end	0		–6		ns	
TWEZMGH	Write enable tristate before **MemGranted**	0		0		ns	2
TWEVMGL	Write enable active after **MemGranted** end	0		–6		ns	

Notes

1 Maximum response time **a** depends on whether an external memory cycle is in progress and whether byte access is active. Maximum time is (2 processor cycles) + (number of wait state cycles) for word access; in byte access mode this time is doubled.

2 When using DMA, **notMemCE** and **notMemWrB0-1** should be pulled up with a resistor (typically 10k). Capacitance should be limited to a maximum of 50pF.

Table 5.6 Memory request

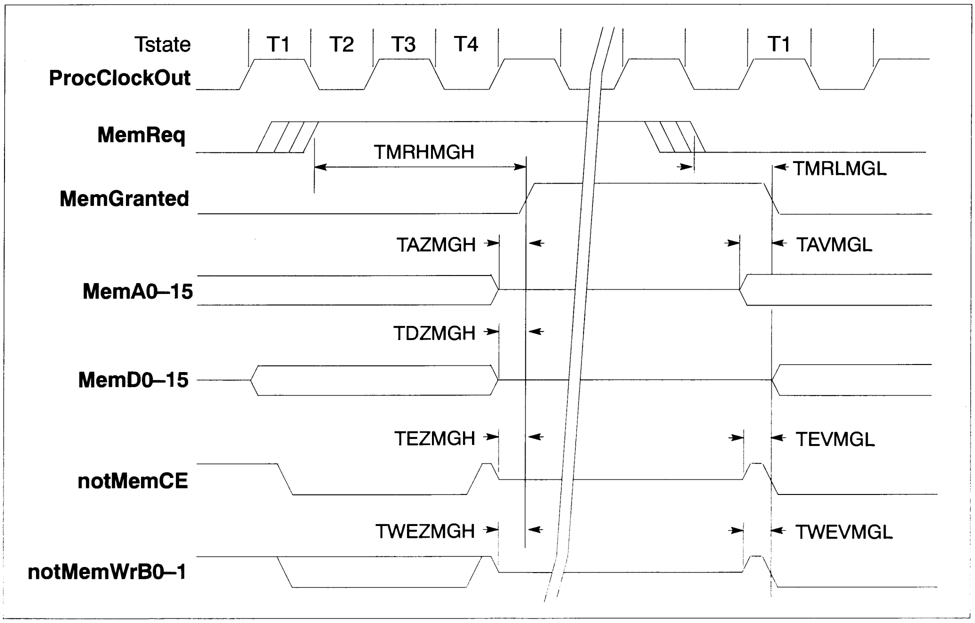


Figure 5.14 IMS T222 memory request timing

6 Events

EventReq and **EventAck** provide an asynchronous handshake interface between an external event and an internal process. When an external event takes **EventReq** high the external event channel (additional to the external link channels) is made ready to communicate with a process. When both the event channel and the process are ready the processor takes **EventAck** high and the process, if waiting, is scheduled. **EventAck** is removed after **EventReq** goes low.

Only one process may use the event channel at any given time. If no process requires an event to occur **EventAck** will never be taken high. Although **EventReq** triggers the channel on a transition from low to high, it must not be removed before **EventAck** is high. **EventReq** should be low during **Reset**; if not it will be ignored until it has gone low and returned high. **EventAck** is taken low when **Reset** occurs.

If the process is a high priority one and no other high priority process is running, the latency is as described on page 70. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

Cycle 1 Sample **EventReq** at pad on the rising edge of **ProcClockOut** and synchronize.

Cycle 2 Edge detect the synchronized **EventReq** and form the interrupt request.

Cycle 3 Sample interrupt vector for microcode ROM in the CPU.

Cycle 4 Execute the interrupt routine for Event rather than the next instruction.

Symbol	Parameter	T222–17		T222-20		Units	Notes
		Min	Max	Min	Max		
TVHKH	**EventReq** response	0		0		ns	1
TKHVL	**EventReq** hold	0		0		ns	1
TVLKL	Delay before removal of **EventAck**	0	178	0	157	ns	
TKLVH	Delay before re-assertion of **EventReq**	0		0		ns	1

Notes

1 Guaranteed, but not tested.

Table 6.1 Event

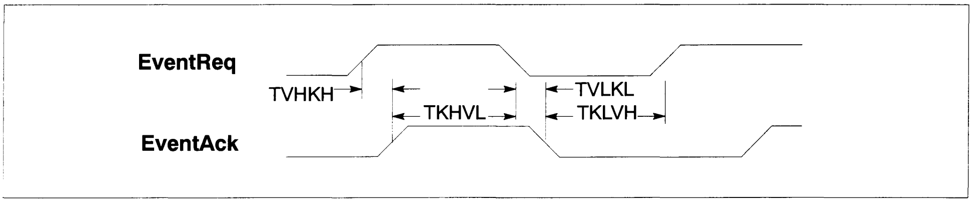


Figure 6.1 IMS T222 event timing

7 Links

Four identical INMOS bi-directional serial links provide synchronized communication between processors and with the outside world. Each link comprises an input channel and output channel. A link between two transputers is implemented by connecting a link interface on one transputer to a link interface on the other transputer. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte. The sending link reschedules the sending process only after the acknowledge for the final byte of the message has been received.

The IMS T222 links allow an acknowledge packet to be sent before the data packet has been fully received. This overlapped acknowledge technique is fully compatible with all other INMOS transputer links. The hard output channels are not double buffered. There is thus a pause between transmission of the last byte of a word of the message and the first byte of the next word. This pause time is related to memory speed. Hard input channels have one byte of double buffering and are unlikely to affect the data rate. The dominant factor affecting link bandwidth is therefore the memory bandwidth of the transmitting transputer, as shown in table 7.1. Internal memory access time is similar to zero wait state external access time. Times are for two interconnected IMS T222's with 20 Mbits/sec link speed.

Memory Speed (20MHz device)	**Byte Output Time ns**	**Word Memory Read ns**	**Unidirectional Data Rate Mbytes/sec**
1 cycle (0 wait)	575	200	1.48
2 cycle (1 wait)	575	250	1.42
3 cycle (2 wait)	575	300	1.38

Table 7.1 Memory/Link speed relationship

The IMS T222 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 5 or 20 Mbits/sec. Links are not synchronised with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

Link speeds can be set by **LinkSpecial**, **Link0Special** and **Link123Special**. The link 0 speed can be set independently. Table 7.2 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed and as **Link123Special** for the others. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

Link Special	Linkn Special	Mbits/sec	Kbytes/sec	
			Uni	Bi
0	0	10	800	1130
0	1	5	430	590
1	0	10	800	1130
1	1	20	1480	2050

Table 7.2 Speed Settings for Transputer Links

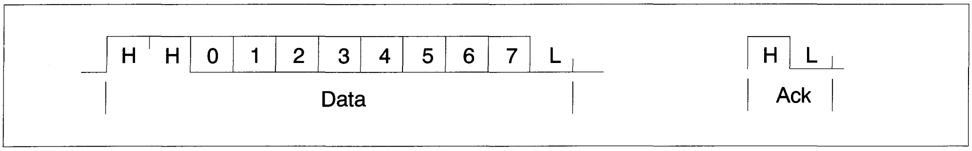


Figure 7.1 IMS T222 link data and acknowledge packets

Symbol	Parameter	Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time			20	ns	1
TJQf	**LinkOut** fall time			10	ns	1
TJDr	**LinkIn** rise time			20	ns	2
TJDf	**LinkIn** fall time			20	ns	2
TJQJD	Buffered edge delay	0			ns	
TJBskew	Variation in TJQJD 20 Mbits/s			3	ns	3
	10 Mbits/s			10	ns	3
	5 Mbits/s			30	ns	3
CLIZ	**LinkIn** capacitance @ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance			50	pF	
RM	Series resistor for 100Ω transmission line		56		ohms	

Notes

1 This parameter is sampled but is not 100% tested.

2 This parameter is not tested.

3 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 7.3 Link

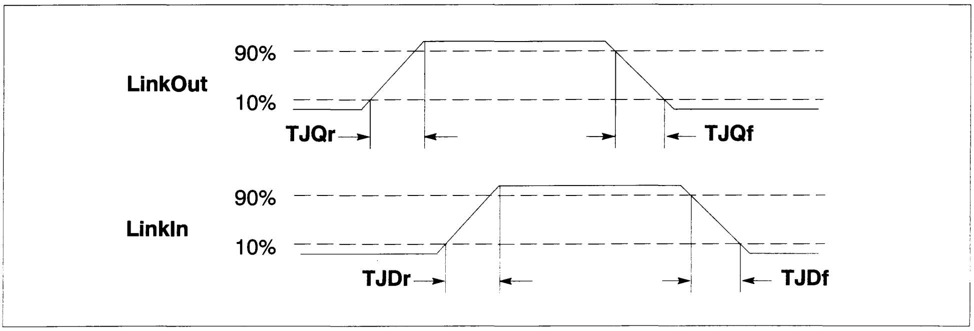


Figure 7.2 IMS T222 link timing

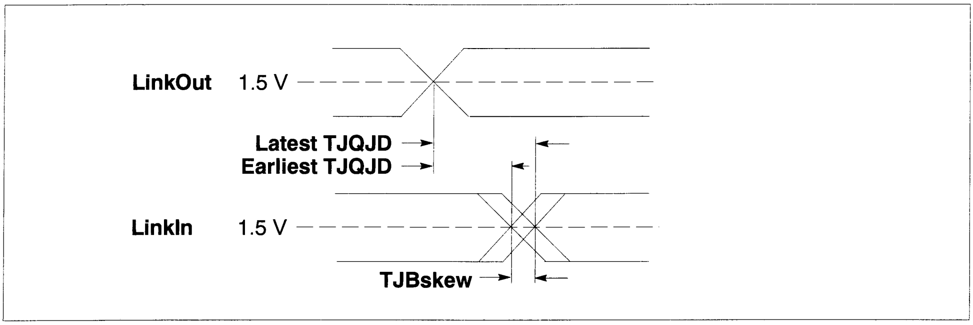


Figure 7.3 IMS T222 buffered link timing

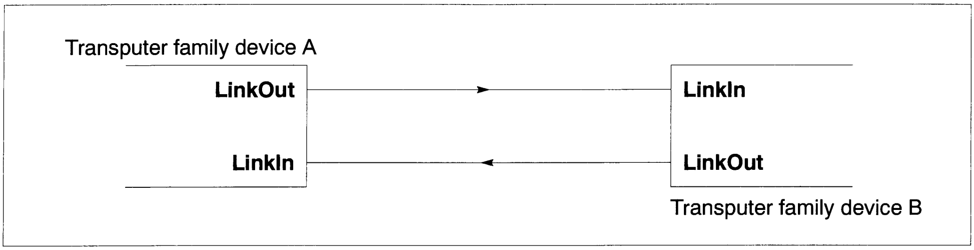


Figure 7.4 Links directly connected

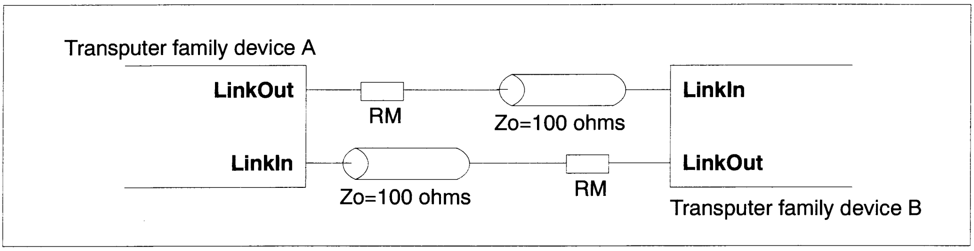


Figure 7.5 Links connected by transmission line

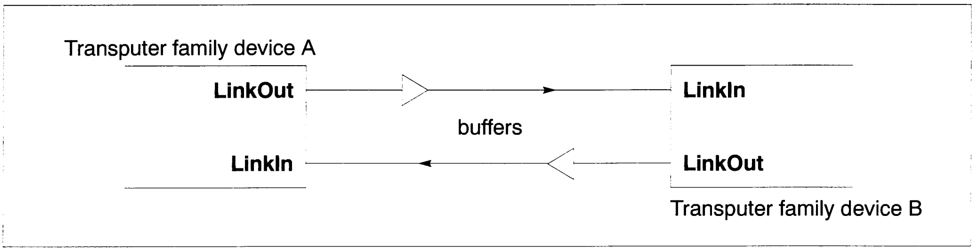


Figure 7.6 Links connected by buffers

8 Electrical specifications

8.1 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
II	Input current		±25	mA	4
tOSC	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 8.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1,2
CL	Load capacitance on any pin		60	pF	
TA	Operating temperature range IMS T222–S	0	70	°C	3

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Air flow rate 400 linear ft/min transverse air flow.

Table 8.2 Operating conditions

SYMBOL	PARAMETER		MIN	MAX	UNITS	NOTES
VIH	High level input voltage		2.0	VDD+0.5	V	1,2
VIL	Low level input voltage		–0.5	0.8	V	1,2
II	Input current	@ GND<VI<VDD		±10	μA	1,2
VOH	Output high voltage	@ IOH=2mA	VDD–1		V	1,2
VOL	Output low voltage	@ IOL=4mA		0.4	V	1,2
IOZ	Tristate output current	@ GND<V0<VDD		±10	μA	1,2
PD	Power dissipation			700	mW	2,3
CIN	Input capacitance	@ f=1MHz		7	pF	4
COZ	Output capacitance	@ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

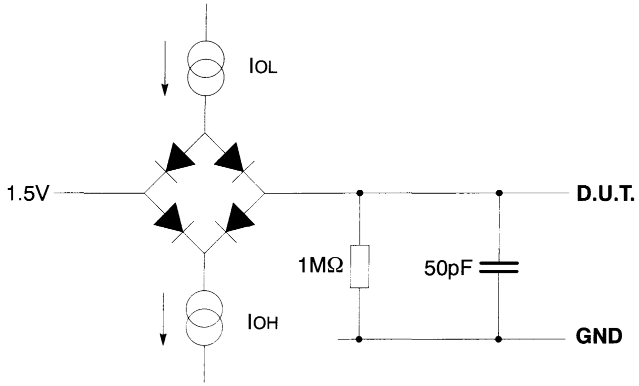
2 Parameters for IMS T222-S measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C. Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution.

4 This parameter is sampled and not 100% tested.

Table 8.3 DC characteristics

8.2 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 8.1 Load circuit for AC measurements

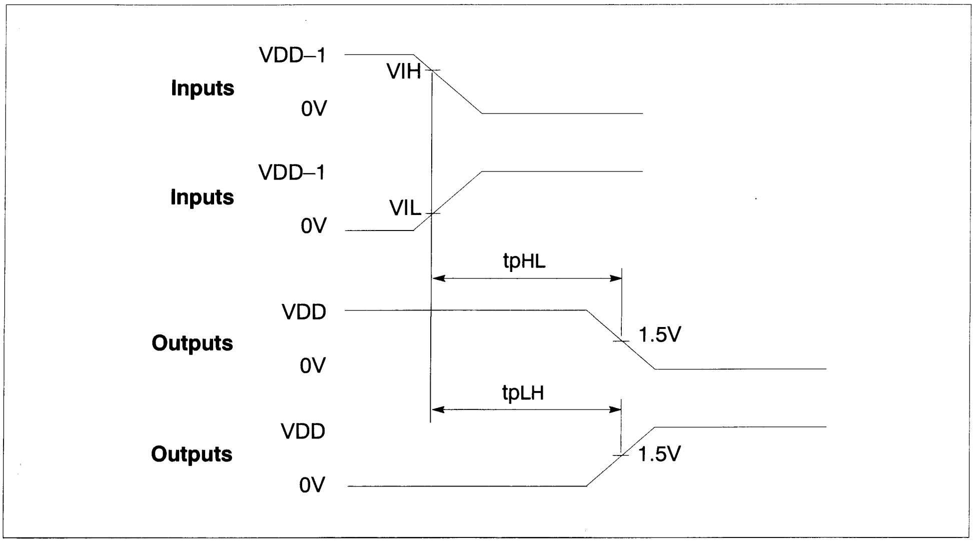


Figure 8.2 AC measurements timing waveforms

8.3 AC timing characteristics

Symbol	Parameter	Min	Max	Units	Notes
TDr	Input rising edges	2	20	ns	1,2,3
TDf	Input falling edges	2	20	ns	1,2,3
TQr	Output rising edges		25	ns	1,4
TQf	Output falling edges		15	ns	1,4

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 This parameter is not tested.

4 This parameter is sampled and is not 100% tested.

Table 8.4 Input and output edges

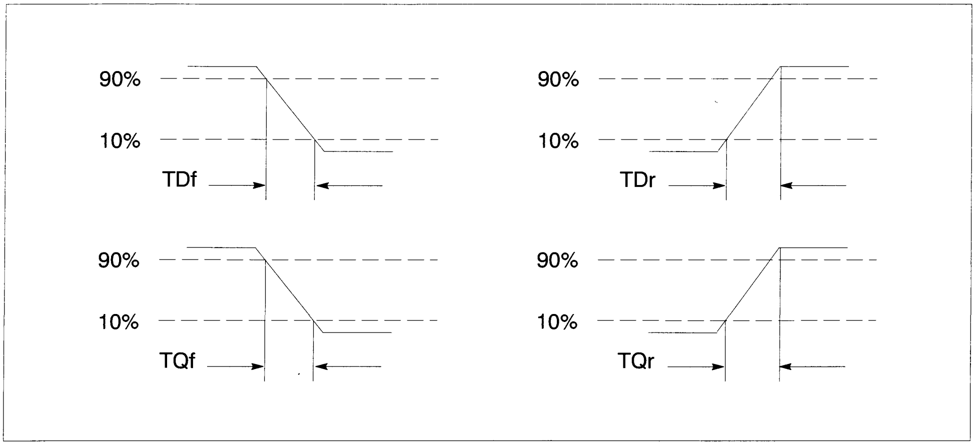


Figure 8.3 IMS T222 input and output edge timing

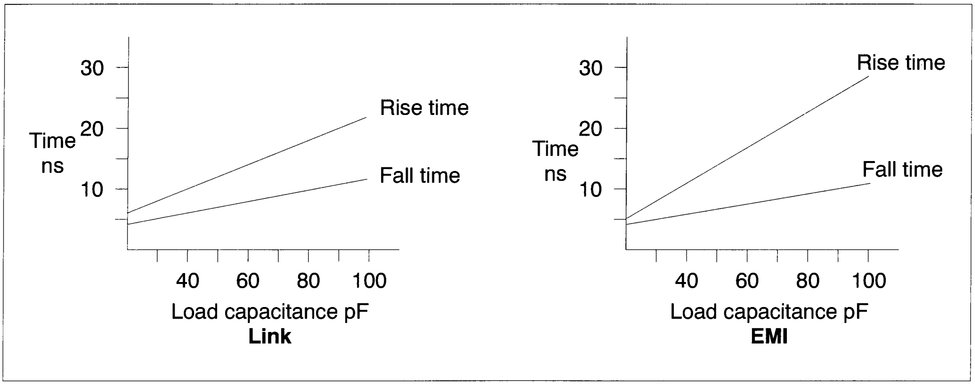


Figure 8.4 Typical rise/fall times

8.4 Power rating

Internal power dissipation (P_{INT}) of transputer and peripheral chips depends on **VDD**, as shown in figure 8.5. P_{INT} is substantially independent of temperature.

Total power dissipation (P_D) of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

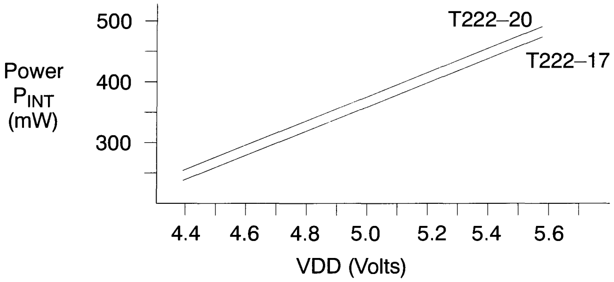
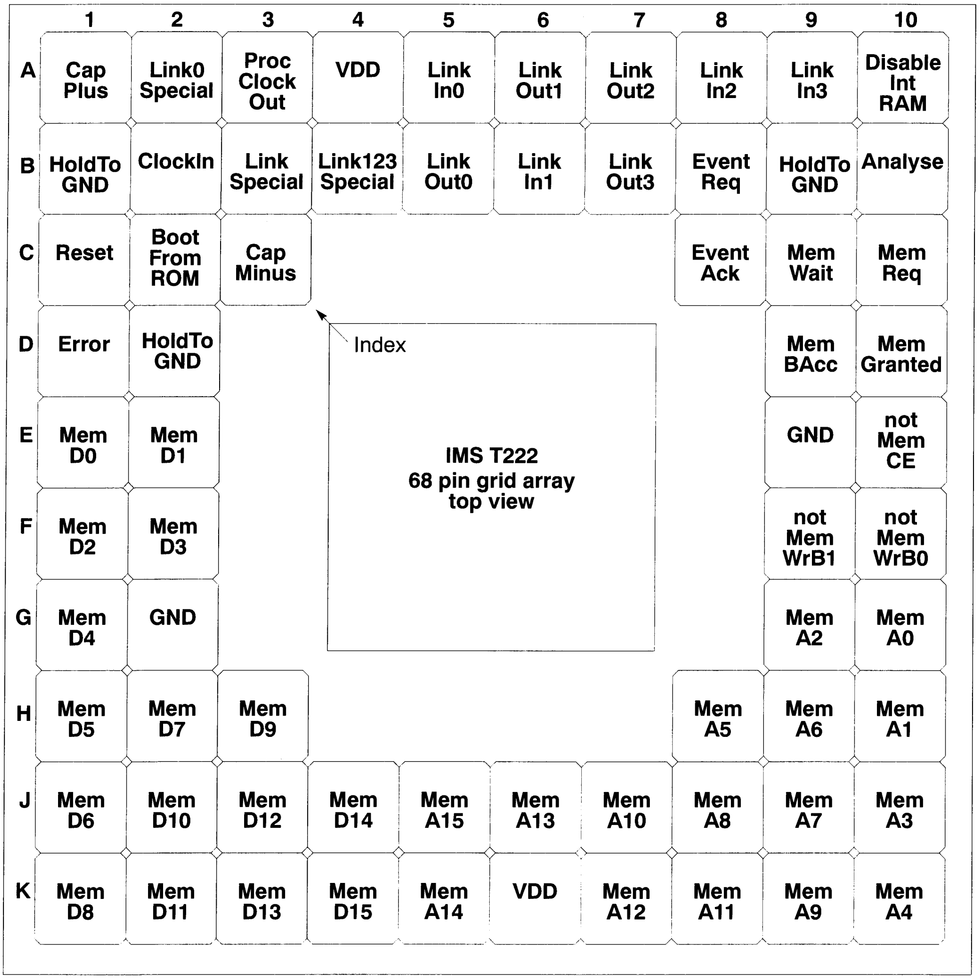


Figure 8.5 IMS T222 internal power dissipation vs VDD

9 Package pinouts

9.1 68 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Cap Plus	Link0 Special	Proc Clock Out	VDD	Link In0	Link Out1	Link Out2	Link In2	Link In3	Disable Int RAM
B	HoldTo GND	ClockIn	Link Special	Link123 Special	Link Out0	Link In1	Link Out3	Event Req	HoldTo GND	Analyse
C	Reset	Boot From ROM	Cap Minus					Event Ack	Mem Wait	Mem Req
D	Error	HoldTo GND							Mem BAcc	Mem Granted
E	Mem D0	Mem D1							GND	not Mem CE
F	Mem D2	Mem D3							not Mem WrB1	not Mem WrB0
G	Mem D4	GND							Mem A2	Mem A0
H	Mem D5	Mem D7	Mem D9					Mem A5	Mem A6	Mem A1
J	Mem D6	Mem D10	Mem D12	Mem D14	Mem A15	Mem A13	Mem A10	Mem A8	Mem A7	Mem A3
K	Mem D8	Mem D11	Mem D13	Mem D15	Mem A14	VDD	Mem A12	Mem A11	Mem A9	Mem A4

Figure 9.1 IMS T222 68 pin grid array package pinout

9.2 68 pin PLCC J-bend package

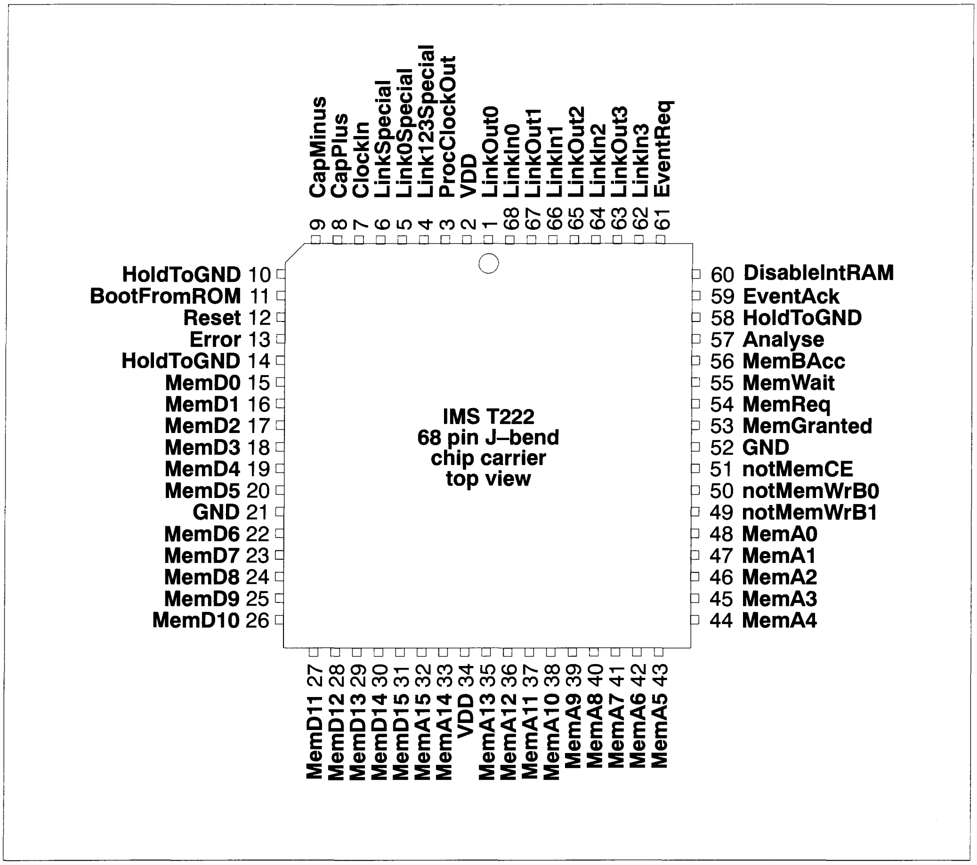


Figure 9.2 IMS T222 68 pin PLCC J-bend package pinout

10 Ordering

This section indicates the designation of speed and package selections for the various devices. Speed of **ClockIn** is 5 MHz for all parts. Transputer processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in the external memory interface section 5.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	Processor clock speed	Processor cycle time	PLLx	Package
IMS T222-G17S	17.5 MHz	57ns	3.5	68 pin ceramic pin grid array
IMS T222-G20S	20.0 MHz	50ns	4.0	68 pin ceramic pin grid array
IMS T222-J17S	17.5 MHz	57ns	3.5	68 pin PLCC J–bend
IMS T222-J20S	20.0 MHz	50ns	4.0	68 pin PLCC J–bend

Table 10.1 IMS T222 ordering details

Military versions to MIL-STD-883 are available, see *'The Military and Space Transputer Databook'* for details.

IMS C004 programmable link switch

inmos®

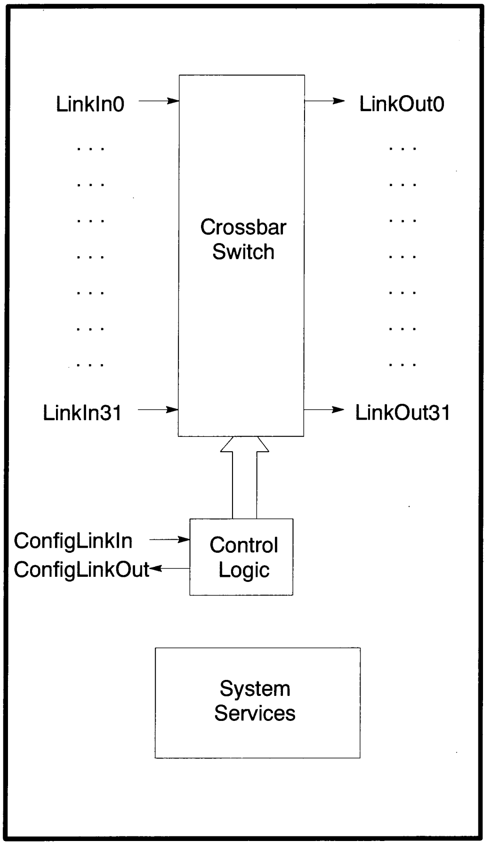
Engineering Data

FEATURES

Standard INMOS serial links
32 way crossbar switch
Regenerates input signal
Cascadable to any depth
No loss of signal integrity
10 or 20 Mbits/sec operating speed
Separate INMOS configuration link
Single +5V ±5% power supply
TTL and CMOS compatibility
1W power dissipation
Packaging 84 pin PGA
MIL-STD-883 device is available

APPLICATIONS

Programmable crossbar switch
Component of larger switch
Reconfigurable supercomputers
Message routing system
High speed multiprocessor systems
Telecommunications
Robotics
Fault tolerant systems
Additional links for transputers



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992

42 1409 03

1 Introduction

The INMOS communication link is a high speed system interconnect which provides full duplex communication between members of the INMOS transputer family, according to the INMOS serial link protocol. The IMS C004, a member of this family, is a transparent programmable link switch designed to provide a full crossbar switch between 32 link inputs and 32 link outputs.

The IMS C004 will switch links running at either the standard speed of 10 Mbits/sec or at the higher speed of 20 Mbits/sec. It introduces, on average, only a 1.75 bit time delay on the signal. Link switches can be cascaded to any depth without loss of signal integrity and can be used to construct reconfigurable networks of arbitrary size. The switch is programmed via a separate serial link called the *configuration link*.

All INMOS products which use communication links, regardless of device type, support a standard communications frequency of 10 Mbits/sec; most products also support 20 Mbits/sec. Products of different type or performance can, therefore, be interconnected directly and future systems will be able to communicate directly with those of today.

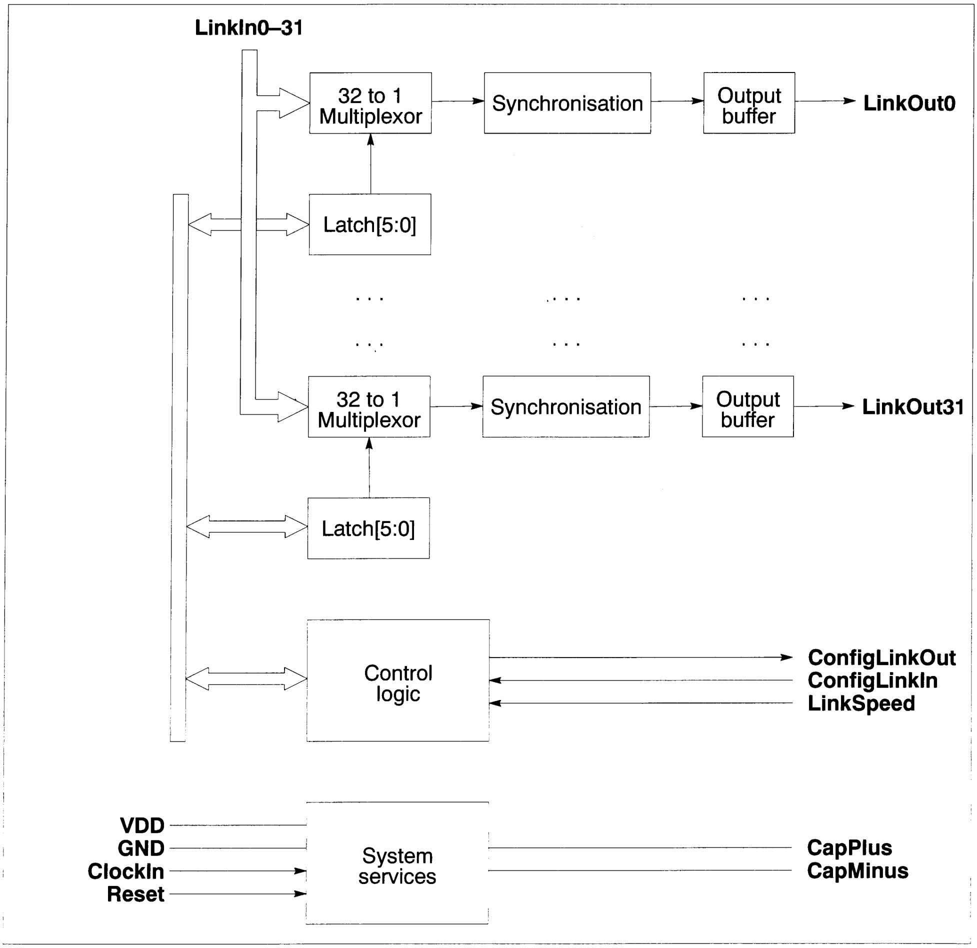


Figure 1.1 IMS C004 block diagram

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high. Pinout details for various packages are given on page 415.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapPlus, CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
Reset	in	System reset
DoNotWire		Must not be wired

Figure 2.1 IMS C004 system services

Pin	In/Out	Function
ConfigLinkIn	in	INMOS configuration link input
ConfigLinkOut	out	INMOS configuration link output

Figure 2.2 IMS C004 configuration

Pin	In/Out	Function
LinkIn0-31	in	INMOS link inputs to the switch
LinkOut0-31	out	INMOS link outputs from the switch
LinkSpeed	in	Link speed selection

Figure 2.3 IMS C004 link

3 System services

System services include all the necessary logic to start up and maintain the IMS C004.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. Several of each are provided to minimise inductance within the package. All supply pins must be connected. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between VDD and GND. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapPlus, CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1μF capacitor to be connected between **CapPlus** and **CapMinus.** A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100kHz and 10MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The connections must not touch power supplies or other noise sources.

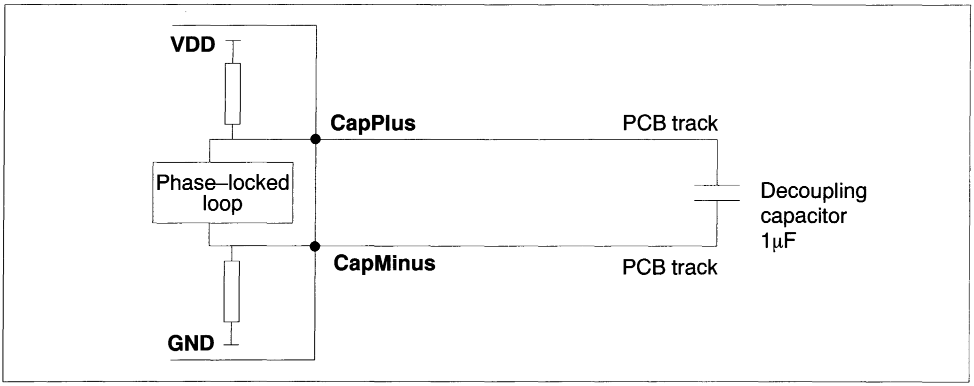


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer family devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1, 2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1., 3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1, 4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 7.3).

Table 3.1 Input clock

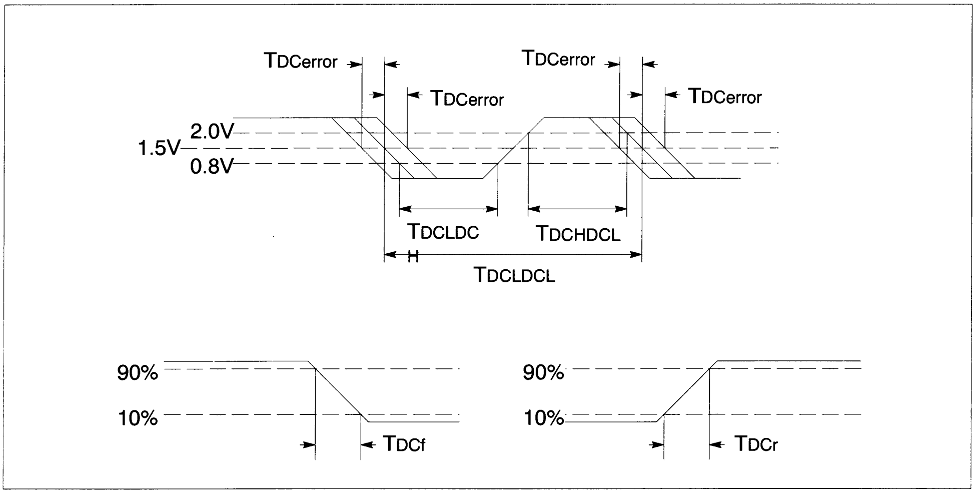


Figure 3.2 **ClockIn** timing

3.4 Reset

The **Reset** pin can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**.

Reset initialises the IMS C004 to a state where all link outputs from the switch are disconnected and held low; the control link is then ready to receive a configuration message.

Symbol	Parameter	Min	Nom	Max	Unit	Notes
TPVRH	Power valid before Reset	10			ms	
TRHRL	Reset pulse width high	8			**ClockIn**	1
TDCVRL	ClockIn running before Reset end	10			ms	2

Notes

1 Full periods of **ClockIn** TDCLDCL required.

2 At power-on reset.

Table 3.2 Reset

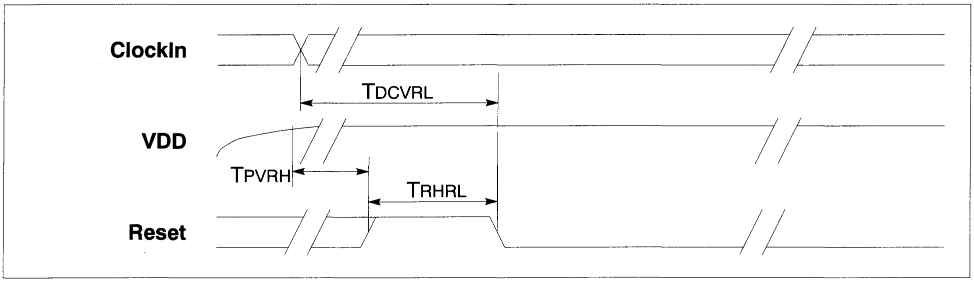


Figure 3.3 Reset timing

4 Links

INMOS bi-directional serial links provide synchronized communication between products and with the outside world. Each link comprises an input channel and output channel. A link between two devices is implemented by connecting a link interface on one to a link interface on the other device. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

A receiver can transmit an acknowledge as soon as it starts to receive a data byte. In this way the transmission of an acknowledge can be overlapped with receipt of a data byte to provide continuous transmission of data. This technique is fully compatible with all other INMOS transputer family links.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies hat the receiving link is able to receive another byte.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

The IMS C004 links support the standard INMOS communication speed of 10Mbits/s. In addition they can be used at 20 Mbit/s. When the **LinkSpeed** pin is low, all links operate at the standard 10Mbit/s;when high they operate at 20Mbits/s.

A single IMS C004 inserted between two transputers which fully implement overlapped acknowledges will cause some reduction in data bandwidth, see table 4.2 and figure 4.7.

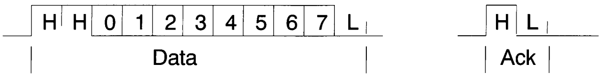


Figure 4.1 IMS C004 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Unit	Notes
TJQr	LinkOut rise time				20	ns	1
TJQf	LinkOut fall time				10	ns	1
TJDr	LinkIn rise time				20	ns	1
TJDf	LinkIn fall time				20	ns	1
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
CLIZ	LinkIn capacitance	@f=1MHz			7	pF	1
CLL	LinkOut load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		Ω	

Notes

1 These parameters are sampled, but are not 100% tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 4.1 Link

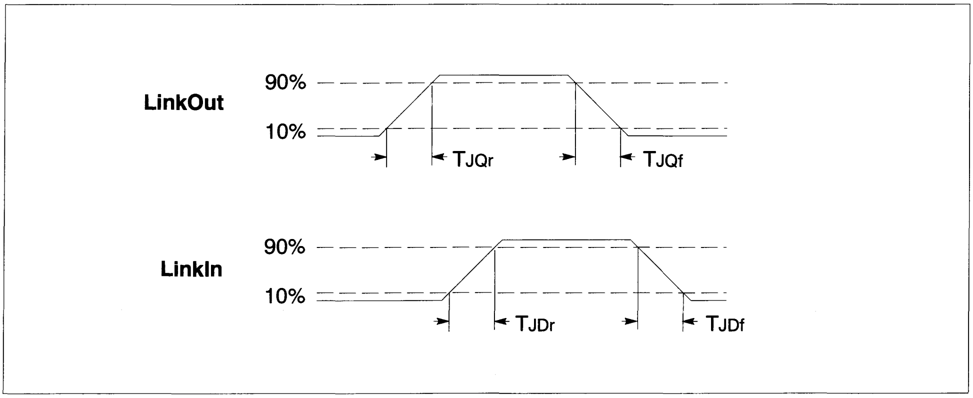


Figure 4.2 IMS C004 link timing

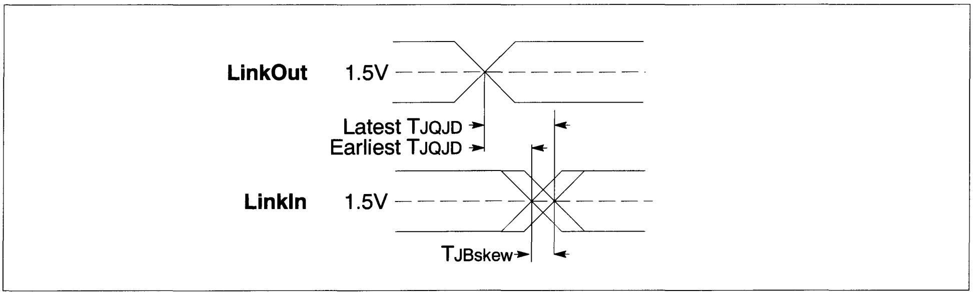


Figure 4.3 IMS C004 buffered link timing

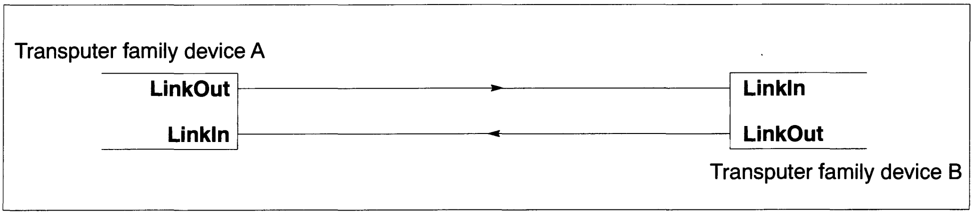


Figure 4.4 IMS C004 links directly connected

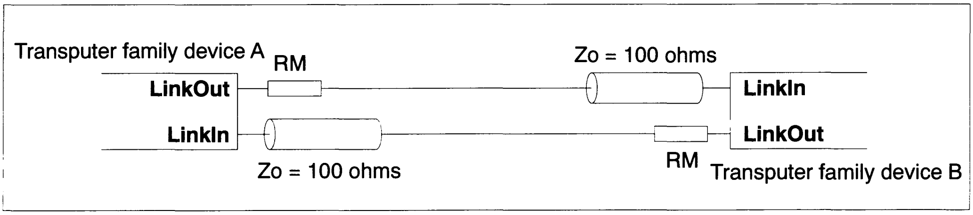


Figure 4.5 IMS C004 links connected by transmission line

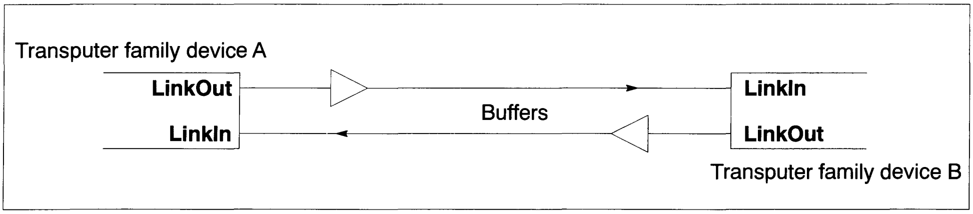
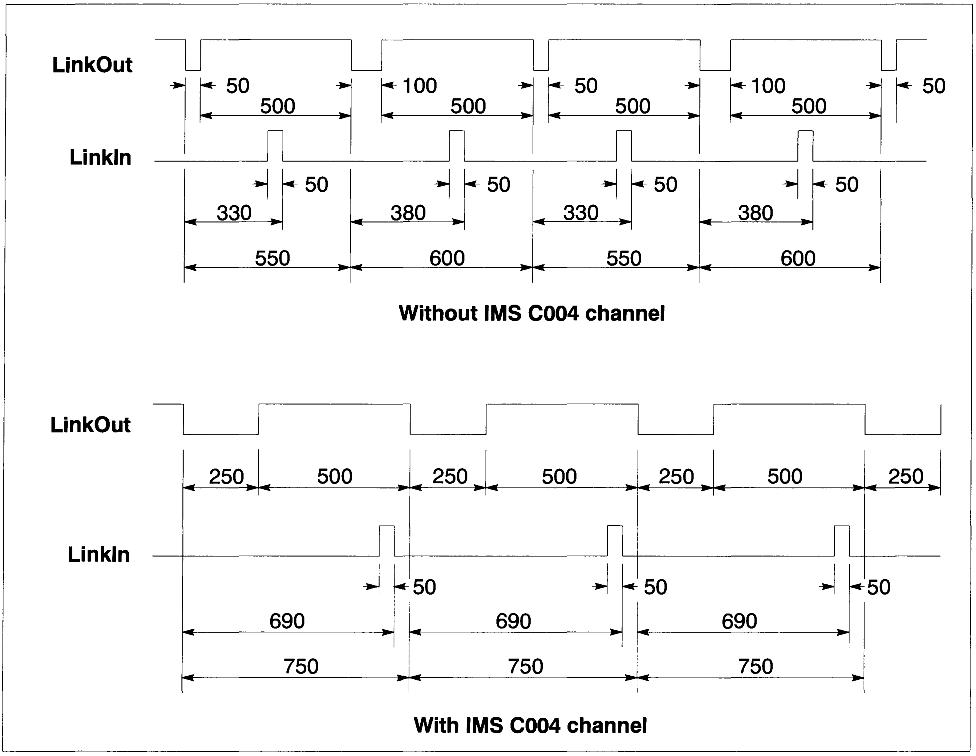


Figure 4.6 IMS C004 links connected by buffers

	Without C004	With C004	Degradation
Unidirectional	1.7	1.3	25%
Biderectional	2.3	2.1	10%

Table 4.2 T800 links data transfer rate at 20Mbit/s



Notes

1 All values are in ns

2 Timing values shown are for links at 20 Mbits/s

Figure 4.7 IMS C004 link timing

5 Switch implementation

The IMS C004 is internally organised as a set of thirty-two 32 to 1 multiplexors. Each multiplexor has associated with it a six bit latch, five bits of which select one input as the source of data for the corresponding output. The sixth bit is used to connect and disconnect the output. These latches can be read and written by messages sent on the configuration link via **ConfigLinkIn** and **ConfigLinkOut**.

The output of each multiplexor is synchronised with an internal high speed clock and regenerated at the output pad. This synchronisation introduces, on average, a 1.75 bit time delay on the signal. As the signal is not electrically degraded in passing through the switch, it is possible to form links through an arbitrary number of link switches.

Each input and output is identified by a number in the range 0 to 31. A configuration message consisting of one, two or three bytes is transmitted on the configuration link. The configuration messages sent to the switch on this link are shown in table 5.1. If an unspecified configuration message is used, the effect of it is undefined.

Configuration Message	Function
[0] [input] [output]	Connects **input** to **output**
[1] [link1] [link2]	Connects **link1** to **link2** by connecting the input of **link1** to the output of **link2** and the input of **link2** to the output of **link1**.
[2] [output]	Enquires which input the **output** is connected to. The IMS C004 responds with the input. The most significant bit of this byte indicates whether the output is connected (bit set high) or disconnected (bit set low).
[3]	This command byte must be sent at the end of every configuration sequence which sets up a connection. The IMS C004 is then ready to accept data on the connected inputs.
[4]	Resets the switch. All outputs are disconnected and held low. This also happens when **Reset** is applied to the IMS C004.
[5] [output]	Output **output** is disconnected and held low.
[6] [link1] [link2]	Disconnects the output of **link1** and the output of **link2**.

Table 5.1 IMS C004 configuration messages

6 Applications

6.1 Link switching

The IMS C004 provides full switching capabilities between 32 INMOS links. It can also be used as a component of a larger link switch. For example, three IMS C004s can be connected together to produce a 48 way switch, as shown in figure 6.1. This technique can be extended to the switch shown in figure 6.2.

A fully connected network of 32 INMOS transputers (one in which all four links are used on every transputer) can be completely configured using just four IMS C004s. Figure 6.5 shows the connected transputer network.

In these diagrams each link line shown represents a unidirectional link; i.e. one output to one input. Where a number is also given, that denotes the number of lines.

6.2 Multiple IMS C004 control

Many systems require a number of IMS C004's, each configured via its own configuration link. A simple method of implementing this uses a master IMS C004, as shown in figure 6.3. One of the transputer links is used to configure the master link switch, whilst another transputer link is multiplexed via the master to send configuration messages to any of the other 31 IMS C004 links.

6.3 Bidirectional exchange

Use of the IMS C004 is not restricted to computer configuration applications. The ability to change the switch setting dynamically enables it to be used as a general purpose message router. This may, of course, also find applications in computing with the emergence of the new generation of supercomputers, but a more widespread use may be found as a communication exchange.

In the application shown in figure 6.4, a message into the exchange must be preceded by a destination token *dest*. When this message is passed, the destination token is replaced with a source token so that the receiver knows where the message has come from. The **in.out** device in the diagram and the controller can be implemented easily with a transputer, and the link protocol for establishing communication with these devices can be interfaced with INMOS link adaptors. All messages from **rx[i]** are preceded by the destination output *dest*. On receipt of such a message the **in.out** device requests the controller to connect a bidirectional link path to *dest*. The controller determines what is currently connected to each end of the proposed link. When both ends are free it sets up the IMS C004 and informs both ends of the new link. Note that in this network two channels are placed on each IMS C004 link, one for each direction.

6.4 Bus systems

The IMS C004 can be used in conjunction with the INMOS IMS C011/C012 link adaptors to provide a flexible means of connecting conventional bus based microprocessor systems.

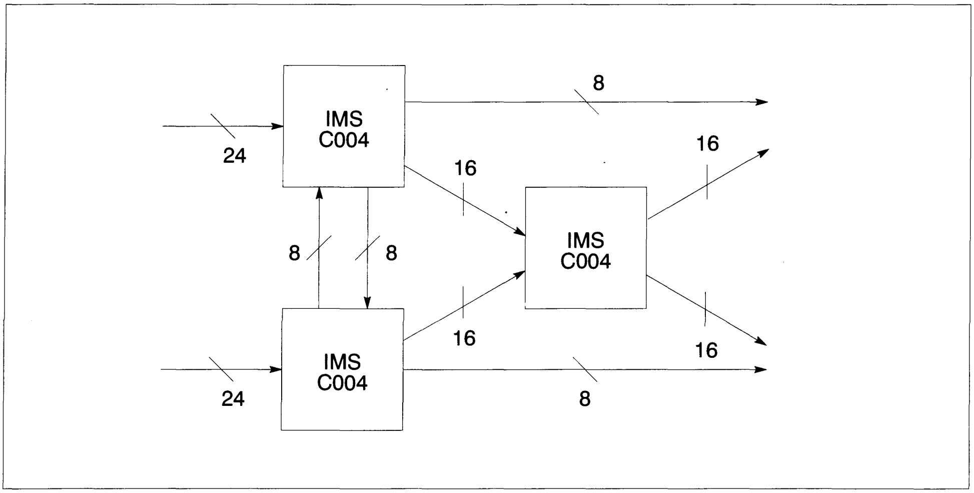


Figure 6.1 48 way link switch

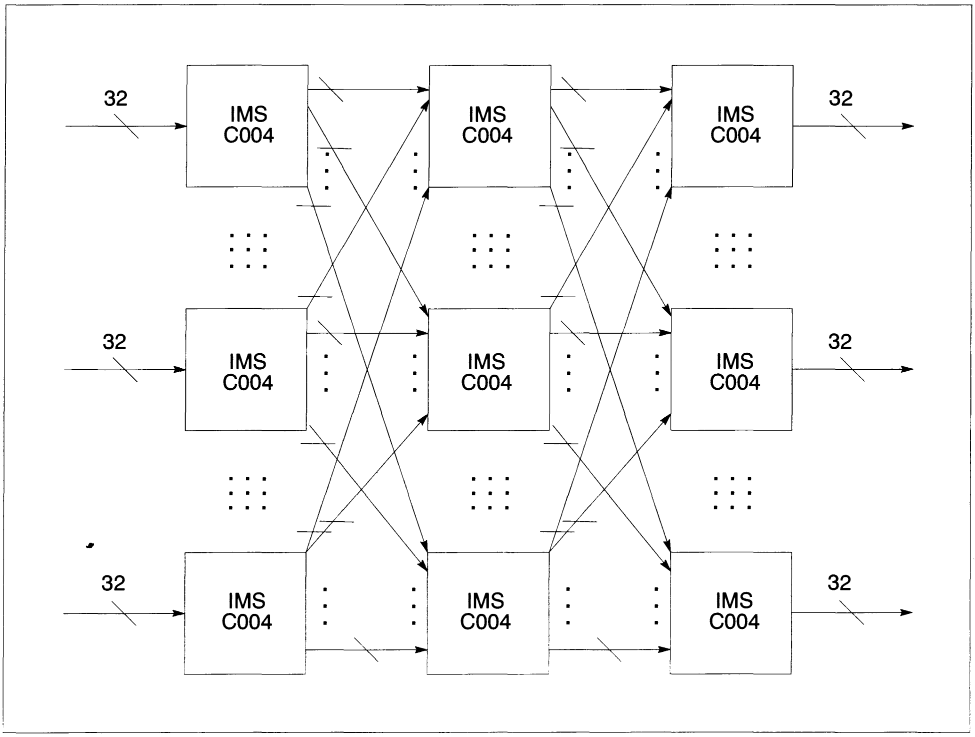


Figure 6.2 Generalised link switch

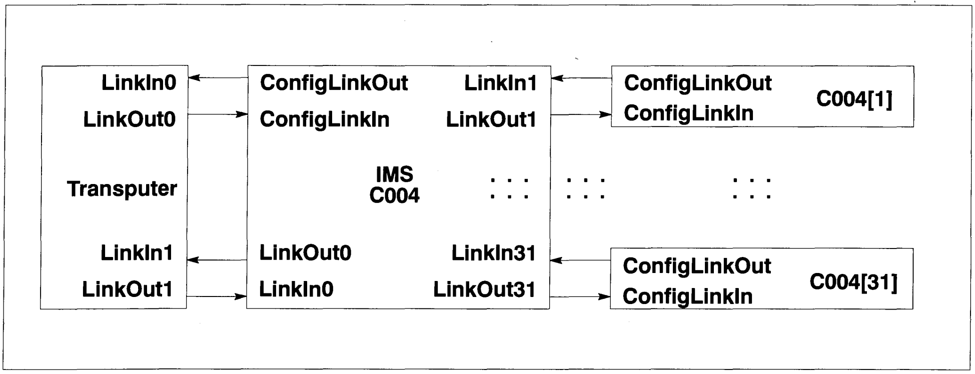


Figure 6.3 Multiple IMS C004 controller

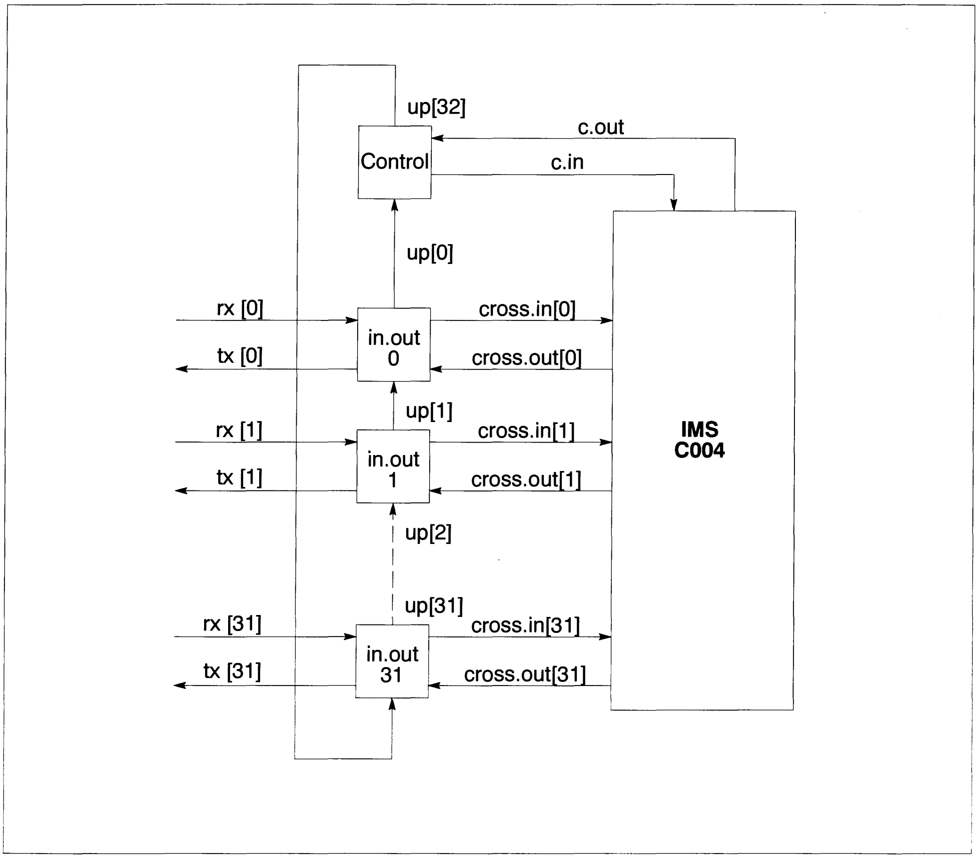


Figure 6.4 32 way bidirectional exchange

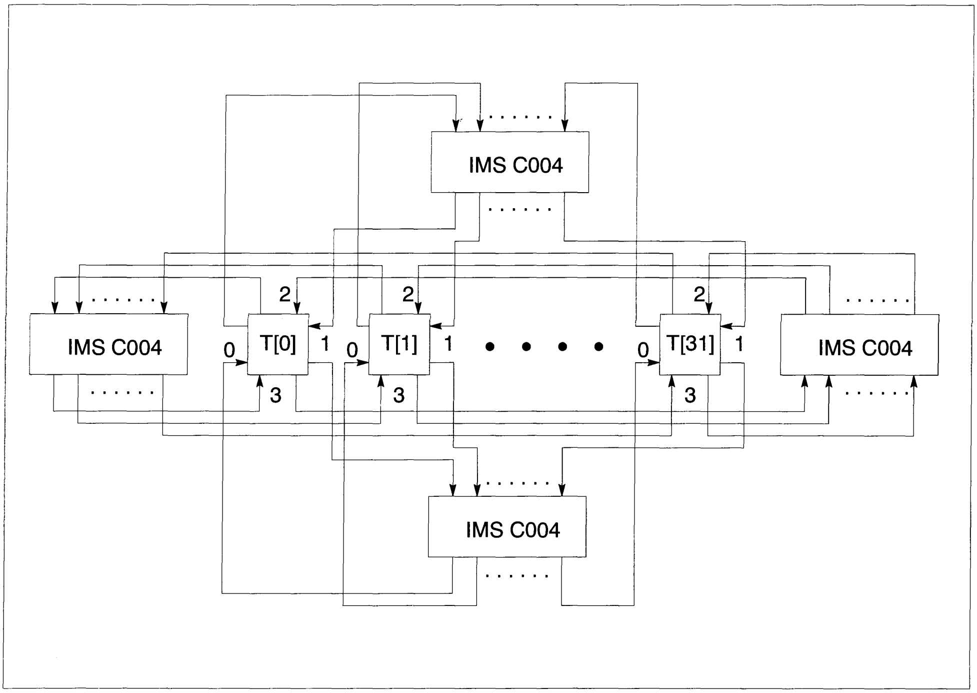


Figure 6.5 Complete connectivity of a transputer network using four IMS C004s

7 Electrical specifications

7.1 DC electrical characteristics

Symbol	Parameter	Min	Max	Unit	Notes
VDD	DC supply voltage	0	7.0	V	1, 2, 3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1, 2, 3
II	Input current		±25	mA	4
OSCT	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		2	W	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 7.1 Absolute maximum ratings

Symbol	Parameter	Min	Max	Unit	Notes
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1,2
CL	Load capacitance on any pin		60	pF	
TA	Operating temperature range IMS C004–S	0	70	°C	3

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Air flow rate 400 linear ft/min transverse air flow.

Table 7.2 Operating conditions

Symbol	Parameter		Min	Max	Unit	Notes
VIH	High level input voltage		2.0	VDD+0.5	V	1,2
VIL	Low level input voltage		–0.5	0.8	V	1,2
II	Input current	@ GND<VI<VDD		±10	μA	1,2
VOH	Output high voltage	@ IOH=2mA	VDD–1		V	1,2
VOL	Output low voltage	@ IOL=4mA		0.4	V	1,2
PD	Power dissipation			1.5	W	2,3
CIN	Input capacitance	@ f=1MHz		7	pF	4
COZ	Output capacitance	@ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

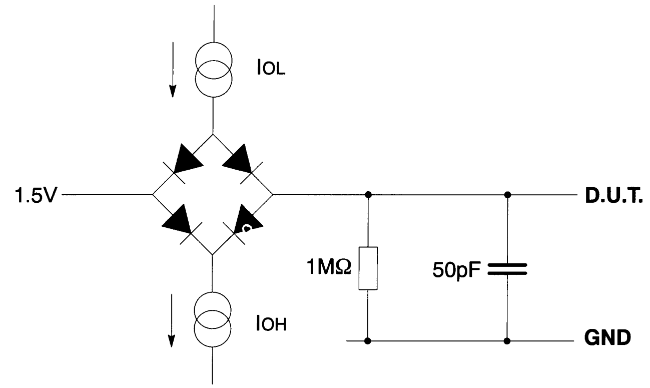
2 Parameters for IMS C004–S measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C. Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and with the number of links active.

4 This parameter is sampled and not 100% tested.

Table 7.3 DC characteristics

7.2 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 7.1 Load circuit for AC measurements

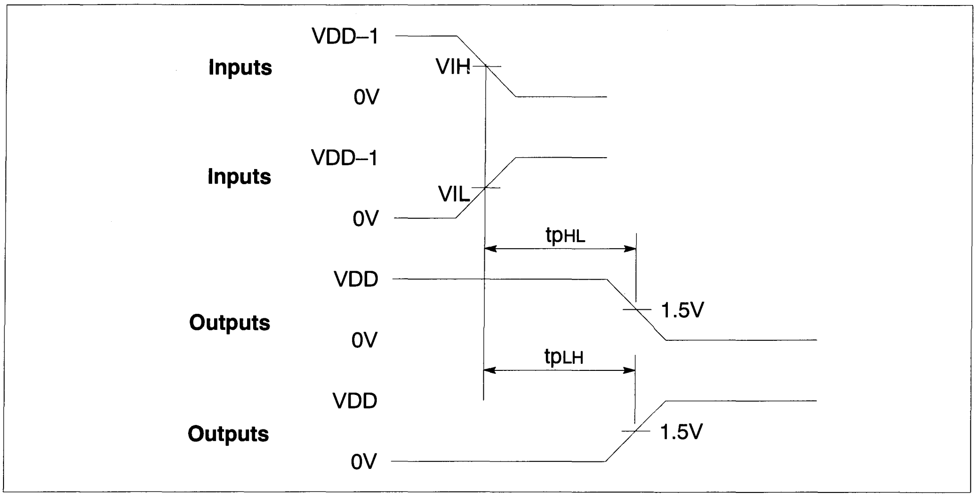


Figure 7.2 AC measurements timing waveforms

7.3 AC timing characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 These parameters are not tested.

Table 7.4 Input and output edges

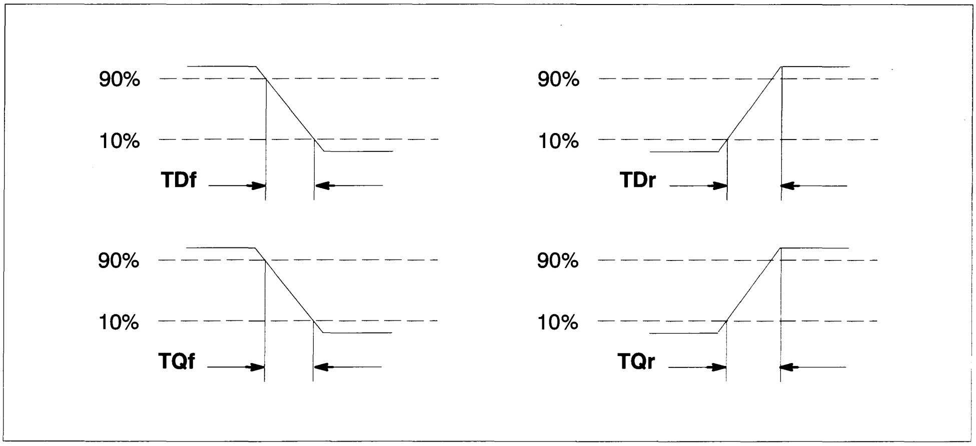


Figure 7.3 IMS C004 input and output edge timing

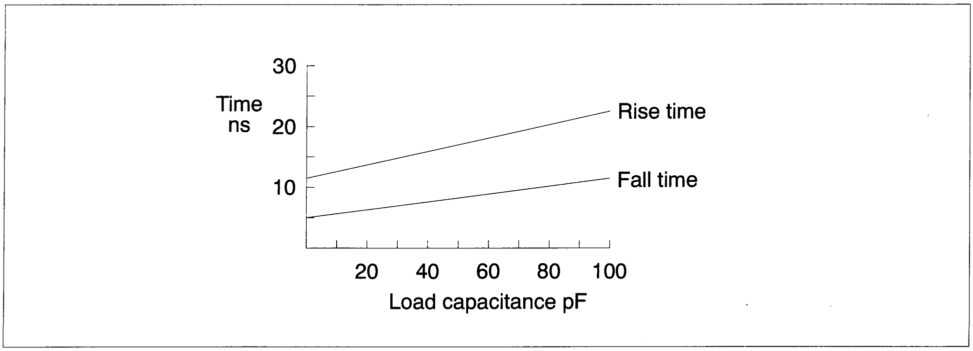


Figure 7.4 Typical link rise/fall times

7.4 Power rating

Internal power dissipation P_{INT} of transputer and peripheral chips depends on VDD, as shown in figure 7.5. P_{INT} is substantially independent of temperature.

Total power dissipation P_D of the chip is:

$$P_D = P_{INT} + P_{IO}$$

Where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is:

$$T_J = T_A + \theta J_A * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

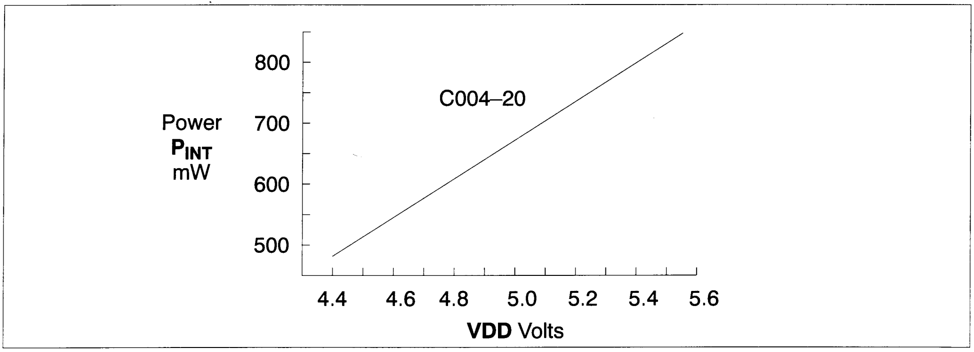
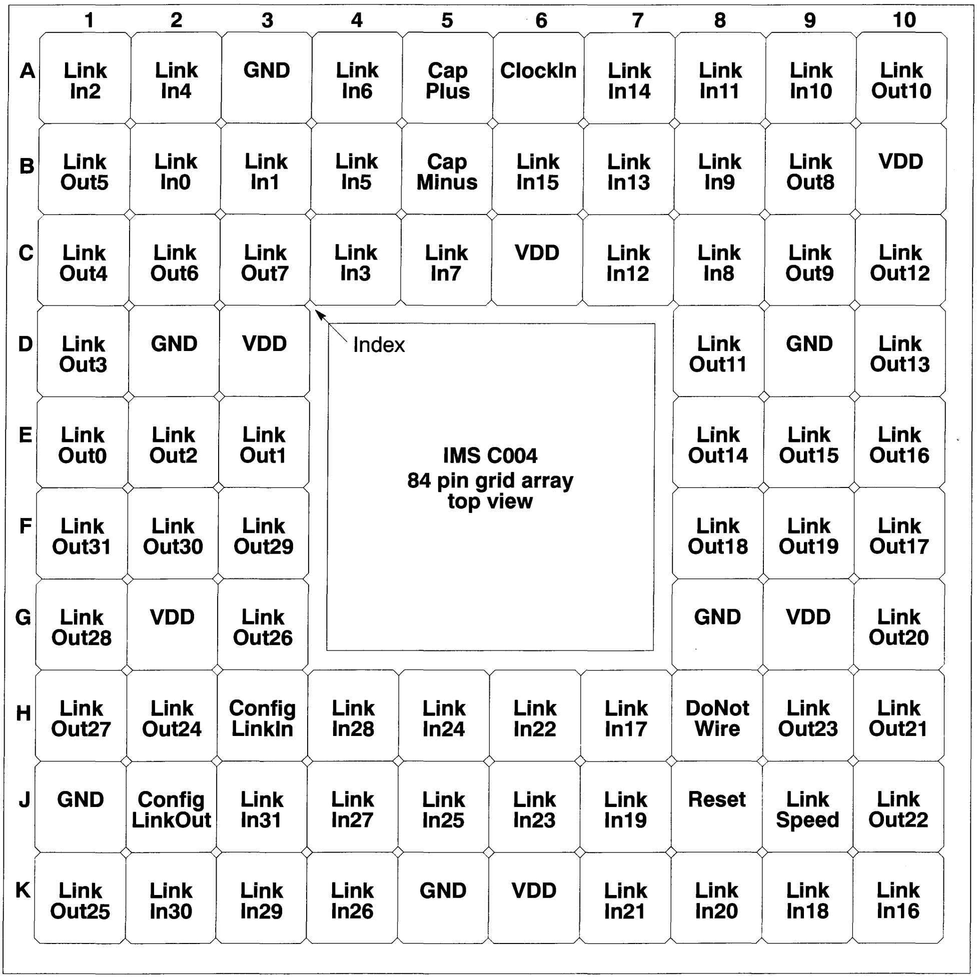


Figure 7.5 IMS C004 internal power dissipation vs VDD

8 Package pinouts

8.1 84 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Link In2	Link In4	GND	Link In6	Cap Plus	ClockIn	Link In14	Link In11	Link In10	Link Out10
B	Link Out5	Link In0	Link In1	Link In5	Cap Minus	Link In15	Link In13	Link In9	Link Out8	VDD
C	Link Out4	Link Out6	Link Out7	Link In3	Link In7	VDD	Link In12	Link In8	Link Out9	Link Out12
D	Link Out3	GND	VDD					Link Out11	GND	Link Out13
E	Link Out0	Link Out2	Link Out1					Link Out14	Link Out15	Link Out16
F	Link Out31	Link Out30	Link Out29					Link Out18	Link Out19	Link Out17
G	Link Out28	VDD	Link Out26					GND	VDD	Link Out20
H	Link Out27	Link Out24	Config LinkIn	Link In28	Link In24	Link In22	Link In17	DoNot Wire	Link Out23	Link Out21
J	GND	Config LinkOut	Link In31	Link In27	Link In25	Link In23	Link In19	Reset	Link Speed	Link Out22
K	Link Out25	Link In30	Link In29	Link In26	GND	VDD	Link In21	Link In20	Link In18	Link In16

Figure 8.1 IMS C004 84 pin grid array

9 Ordering

This section indicates the designation of package selections for the IMS C004. Speed of **ClockIn** is 5 MHz for all parts.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	Package
IMS C004-G20S	84 pin ceramic pin grid array

Table 9.1 IMS C004 ordering details

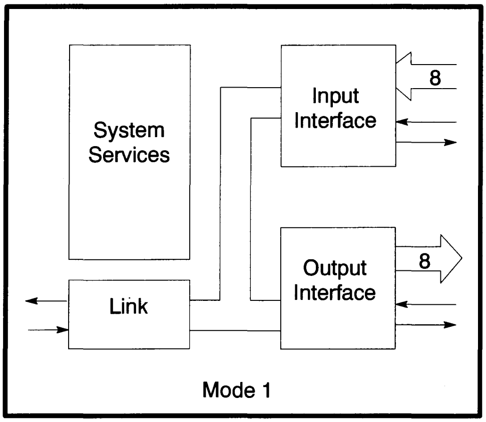
Military versions to MIL-STD-883 are available, see *'The Military and Space Transputer Databook'* for details.

IMS C011 link adaptor

Engineering Data

FEATURES

Standard INMOS link protocol
10 or 20 Mbits/sec operating speed
Communicates with INMOS transputers
Converts between serial link and parallel bus
Converts between serial link and parallel device



Mode 1

Two modes of parallel operation:

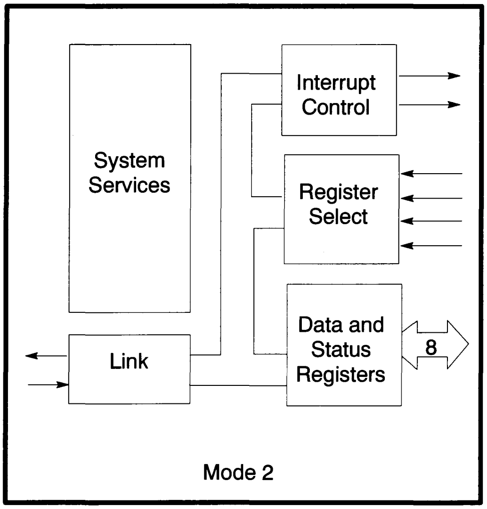
Mode 1: Peripheral interface

Eight bit parallel input interface
Eight bit parallel output interface
Full handshake on input and output

Mode 2: Bus interface

Tristate bidirectional bus interface
Memory mapped registers
Interrupt capability

Single +5V ±5% power supply
TTL and CMOS compatibility
120mW power dissipation
28 pin 0.6” plastic package, 28 pin SOJ package
MIL-STD-883 device is available



Mode 2

APPLICATIONS

Programmable I/O for transputer
Connecting microprocessors to transputers
High speed links between microprocessors
Inter-family microprocessor interfacing
Interconnecting different speed links

SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992
42 1412 05

1 Introduction

The INMOS communication link is a high speed system interconnect which provides full duplex communication between members of the INMOS transputer family, according to the INMOS serial link protocol. The IMS C011, a member of this family, provides for full duplex transputer link communication with standard microprocessor and sub-system architectures, by converting bi-directional serial link data into parallel data streams.

All INMOS products which use communication links, regardless of device type, support a standard communications frequency of 10 Mbits/sec; most products also support 20 Mbits/sec. Products of different type or performance can, therefore, be interconnected directly and future systems will be able to communicate directly with those of today. The IMS C011 link will run at either the standard speed of 10 Mbits/sec or at the higher speed of 20 Mbits/sec. Data reception is asynchronous, allowing communication to be independent of clock phase.

The link adaptor can be operated in one of two modes. In Mode 1 the IMS C011 converts between a link and two independent fully handshaken byte-wide interfaces, one input and one output. It can be used by a peripheral device to communicate with a transputer, an INMOS peripheral processor or another link adaptor, or it can provide programmable input and output pins for a transputer. Two IMS C011 devices in this mode can be connected back to back via the parallel ports and used as a frequency changer between different speed links.

In Mode 2 the IMS C011 provides an interface between an INMOS serial link and a microprocessor system bus. Status and data registers for both input and output ports can be accessed across the byte-wide bi-directional interface. Two interrupt outputs are provided, one to indicate input data available and one for output buffer empty.

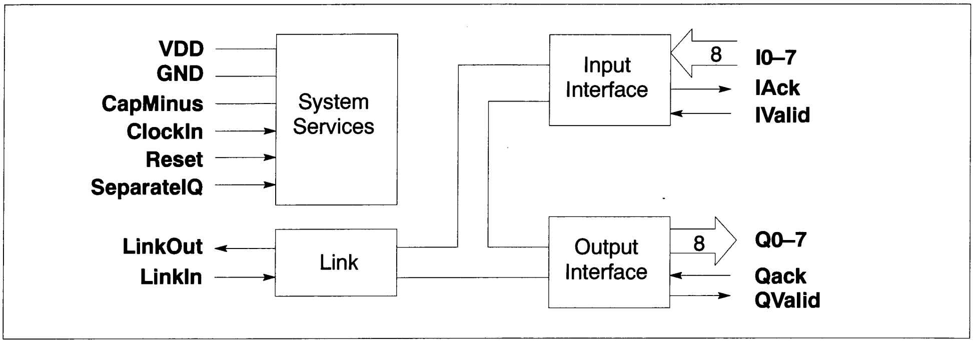


Figure 1.1 IMS C011 Mode 1 block diagram

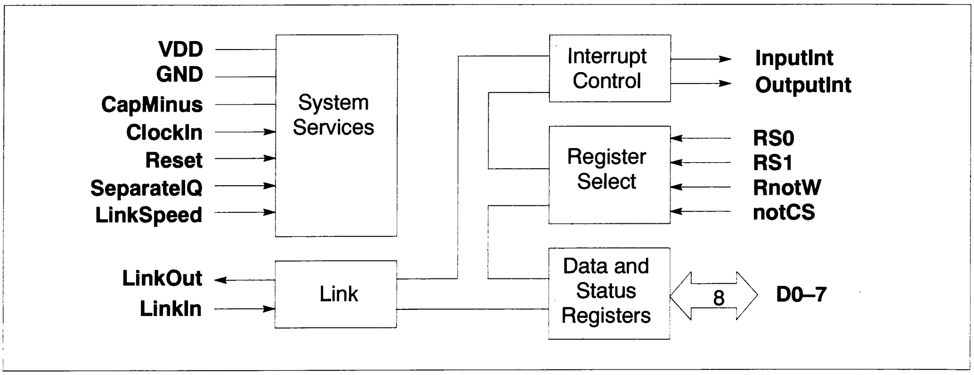


Figure 1.2 IMS C011 Mode 2 block diagram

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high. Pinout details for various packages are given on page 439.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
Reset	in	System reset
SeparateIQ	in	Select mode and Mode 1 link speed
LinkIn	in	Serial data input channel
LinkOut	out	Serial data output channel

Table 2.1 Services and link

Pin	In/Out	Function
I0-7	in	Parallel input bus
IValid	in	Data on **I0-7** is valid
IAck	out	Acknowledge **I0-7** data received by other link
Q0-7	out	Parallel output bus
QValid	out	Data on **Q0-7** is valid
QAck	in	Acknowledge from device: data **Q0-7** was read

Table 2.2 Mode 1 parallel interface

Pin	In/Out	Function
D0-7	in/out	Bi-directional data bus
notCS	in	Chip select
RS0-1	in	Register select
RnotW	in	Read/write control signal
InputInt	out	Interrupt on link receive buffer full
OutputInt	out	Interrupt on link transmit buffer empty
LinkSpeed	in	Select link speed as 10 or 20 Mbits/sec
HoldToGND		Must be connected to **GND**
DoNotWire		Must not be wired

Table 2.3 Mode 2 parallel interface

3 System services

System services include all the necessary logic to start up and maintain the IMS C011.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

AC noise between **VDD** and **GND** must be kept below 200 mV peak to peak at all frequencies above 100 KHz. AC noise between **VDD** and the ground reference of load capacitances must be kept below 200 mV peak to peak at all frequencies above 30 MHz. Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1μF capacitor to be connected between **VDD** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 10 MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The positive connection of the capacitor must be connected directly to **VDD**. Connections must not otherwise touch power supplies or other noise sources.

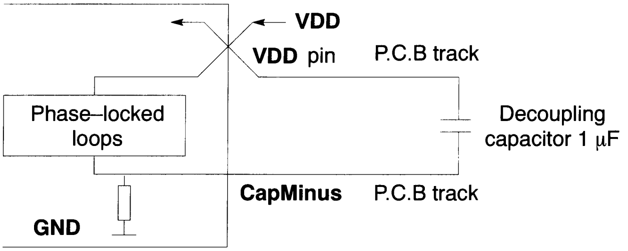


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer family devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200		ns	1, 2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1, 3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1, 4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1 Guaranteed, but not tested.

2 Measured between corresponding points on consecutive falling edges.

3 Variation of individual falling edges from their nominal times.

4 This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.

5 Clock transitions must be monotonic within the range **VIH** to **VIL** (table 7.3).

Table 3.1 Input clock

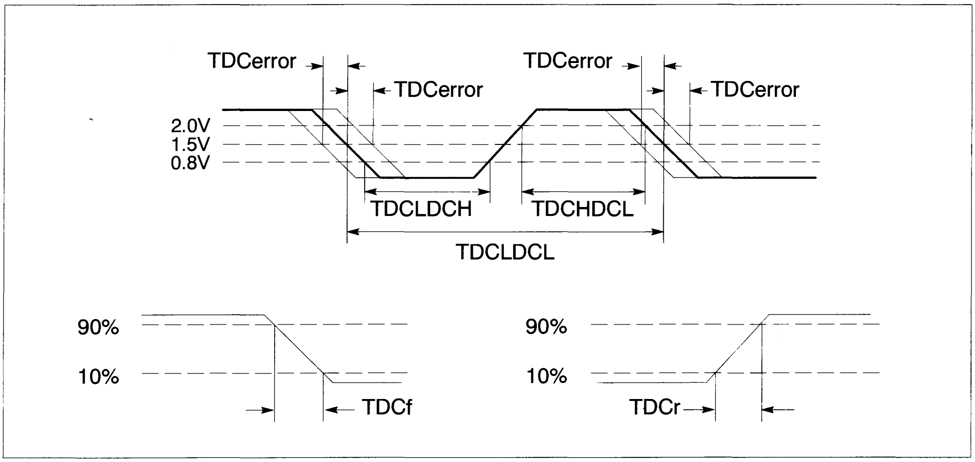


Figure 3.2 ClockIn timing

3.4 SeparateIQ

The IMS C011 link adaptor has two different modes of operation. Mode 1 is basically a link to peripheral adaptor, whilst Mode 2 interfaces between a link and a microprocessor bus system.

Mode 1 can be selected for one of two link speeds by connecting **SeparateIQ** to **VDD** (10 Mbits/sec) or to **ClockIn** (20 Mbits/sec).

Mode 2 is selected by connecting **SeparateIQ** to **GND**; in this mode 10 Mbits/sec or 20 Mbits/sec is selected by **LinkSpeed**. Link speeds are specified for a **ClockIn** frequency of 5 MHz.

In order to select the link speed, **SeparateIQ** may be changed dynamically providing the link is in a quiescent state and no input or output is required. **Reset** must be applied subsequent to the selection to initialise the device. If **ClockIn** is gated to achieve this, its skew must be limited to the value **TDCHSIQH** shown in table 3.3. The mode of operation (Mode 1, Mode 2) must not be changed dynamically.

SeparateIQ	Mode	Link Speed Mbits/sec
VDD	1	10
ClockIn	1	20
GND	2	10 or 20

Table 3.2 **SeparateIQ** mode selection

Symbol	Parameter	Min	Nom	Max	Units	Notes
TDCHSIQH	Skew from **ClockIn** to **ClockIn**			20	ns	1

Notes

1 Skew between **ClockIn** arriving on the **ClockIn** pin and on the **SeparateIQ** pin.

Table 3.3 **SeparateIQ**

3.5 Reset

The **Reset** pin can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. All inputs, with the exception of **ClockIn** and **SeparateIQ** (plus **LinkSpeed** in mode 2), must be held in their inactive state during reset.

Reset initialises the IMS C011 to the following state: **LinkOut** is held low; the control outputs (**IAck** and **QValid** in Mode 1, **InputInt** and **OutputInt** in Mode 2) are held low; interrupts (Mode 2) are disabled; the states of **Q0-7** in Mode 1 are unspecified; **D0-7** in Mode 2 are high impedance.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TPVRH	Power valid before **Reset**	10			ms	1
TRHRL	**Reset** pulse width high	8			ClockIn	1,2
TDCVRL	**ClockIn** running before **Reset** end	10			ms	1,3
TRLIvH	**Reset** low before **IValid** high (mode 1)	0			ns	1
TRLCSL	**Reset** low before chip select low (mode 2)	0			ns	1

Notes

1 This parameter is not tested.

2 Full periods of **ClockIn TDCLDCL** required.

3 At power-on reset.

Table 3.4 Reset

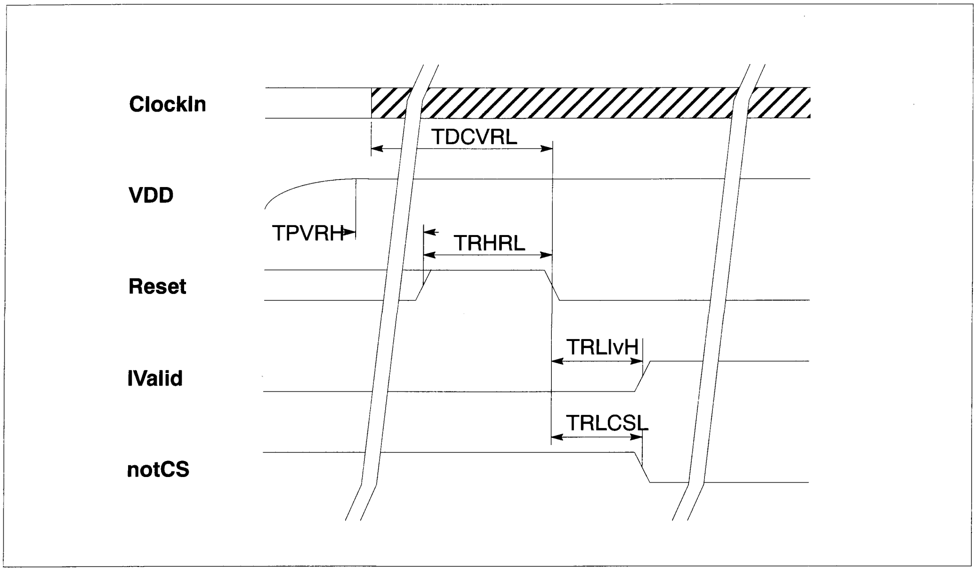


Figure 3.3 **Reset** timing

4 Links

INMOS bi-directional serial links provide synchronized communication between INMOS products and with the outside world. Each link comprises an input channel and output channel. A link between two devices is implemented by connecting a link interface on one device to a link interface on the other device. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte.

Links are not synchronised with **ClockIn** and are insensitive to its phase. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

The IMS C011 link supports the standard INMOS communication speed of 10 Mbits/sec. In addition it can be used at 20 Mbits/sec. Link speed can be selected in one of two ways. In Mode 1 it is altered by **SeparateIQ** (page 421). In Mode 2 it is selected by **LinkSpeed**; when the **LinkSpeed** pin is low, the link operates at the standard 10 Mbits/sec; when high it operates at 20 Mbits/sec.

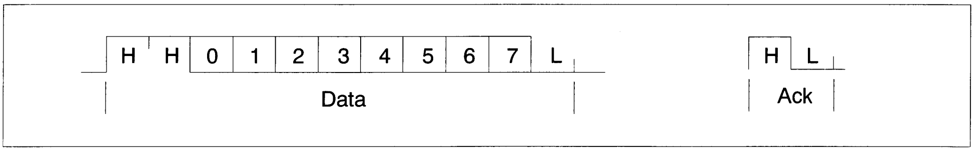


Figure 4.1 IMS C011 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	2
TJDf	**LinkIn** fall time				20	ns	2
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	3
		10 Mbits/s			10	ns	3
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		ohms	

Notes

1 This parameter is sampled but is not 100% tested.

2 This parameter is not tested.

3 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

Table 4.1 Link

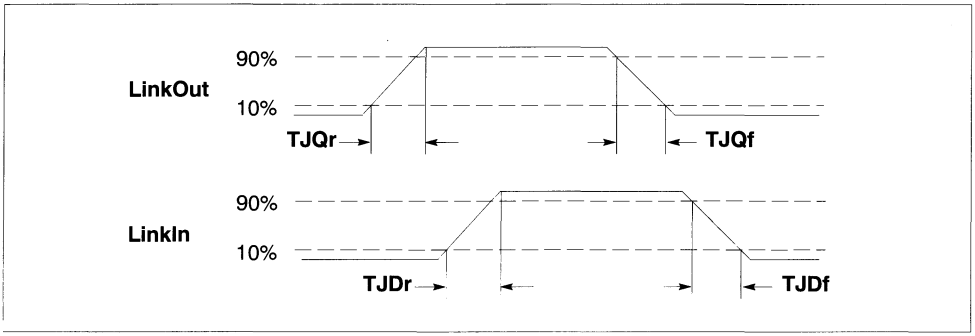


Figure 4.2 IMS C011 link timing

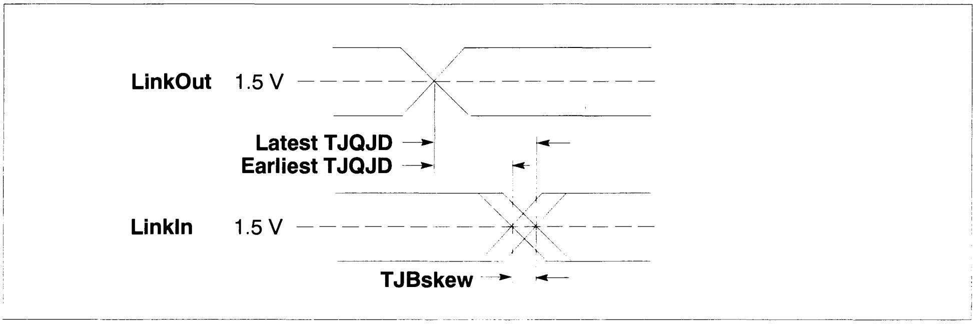


Figure 4.3 IMS C011 buffered link timing

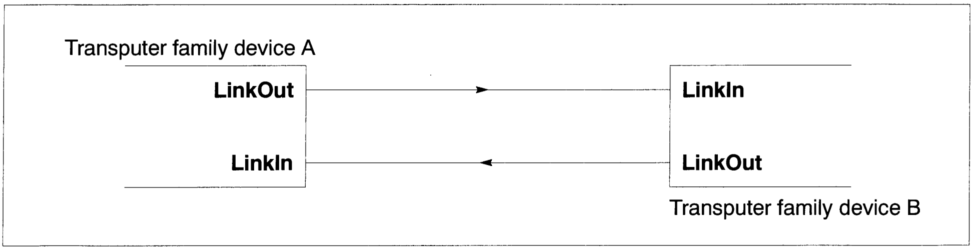


Figure 4.4 Links directly connected

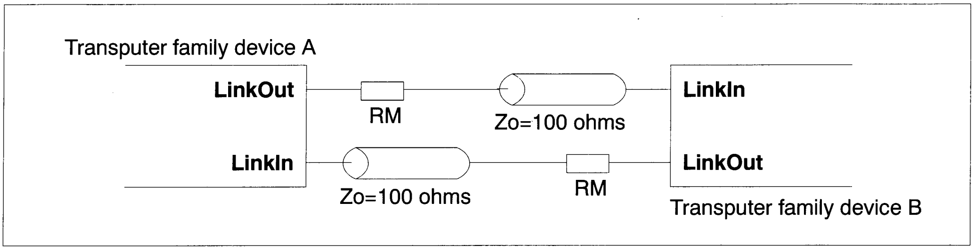


Figure 4.5 Links connected by transmission line

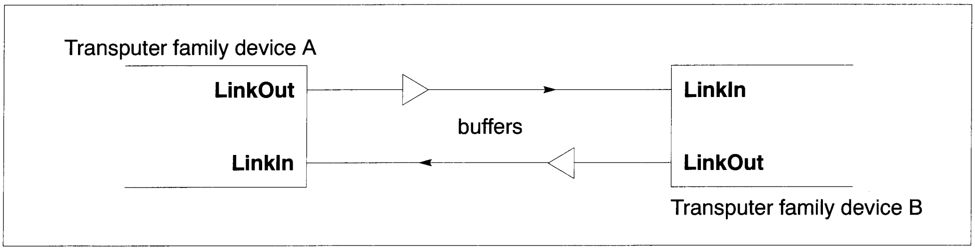


Figure 4.6 Links connected by buffers

5 Mode 1 parallel interface

In Mode 1 the IMS C011 link adaptor is configured as a parallel peripheral interface with handshake lines. Communication with a transputer family device is via the serial link. The parallel interface comprises an input port and an output port, both with handshake.

5.1 Input port

The eight bit parallel input port **I0-7** can be read by a transputer family device via the serial link. **IValid** and **IAck** provide a simple two-wire handshake for this port. When data is valid on **I0-7**, **IValid** is taken high by the peripheral device to commence the handshake. The link adaptor transmits data presented on **I0-7** out through the serial link. After the data byte transmission has been completed and an acknowledge packet is received on the input link, the IMS C011 sets **IAck** high. To complete the handshake, the peripheral device must return **IValid** low. The link adaptor will then set **IAck** low. New data should not be put onto **I0-7** until **IAck** is returned low.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TIdVIvH	Data setup	5			ns	
TIvHLdV	**IValid** high to link data output	0.8		2.5	bits	1,2
TLaVIaH	Link acknowledge start to **IAck** high			3.5	bits	1,3
TIaHIdX	Data hold after **IAck** high	0			ns	
TIaHIvL	**IValid** hold after **IAck** high	0			ns	
TIvLIaL	**IAck** hold after **IValid** low	0.8		3	bits	1
TIaLIvH	Delay before next **IValid** high	0			ns	

Notes

1 Unit of measurement is one link data bit time; at 10 Mbits/s data link speed, one bit time is nominally 100 ns.

2 Maximum time assumes there is no acknowledge packet already on the link. Maximum time with acknowledge on the link is extended by 2 bits.

3 Both data transmission and the returned acknowledge must be completed before **IAck** can go high.

Table 5.1 Mode 1 parallel data input

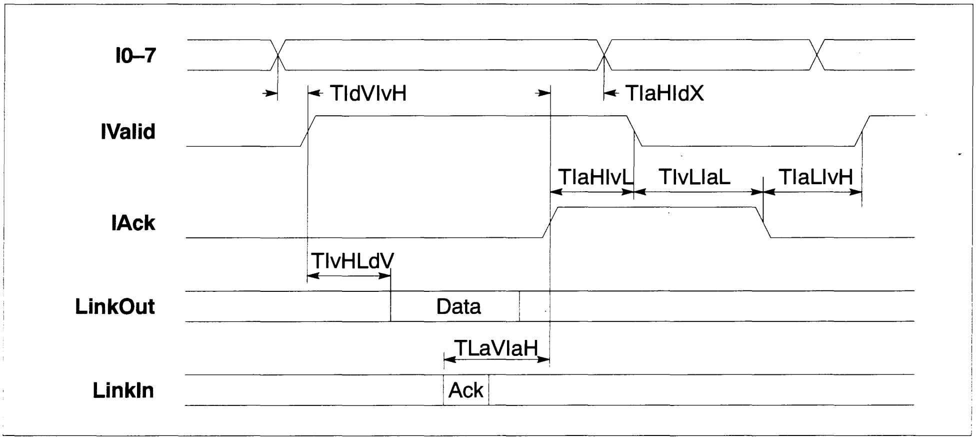


Figure 5.1 IMS C011 Mode 1 parallel data input to link adaptor

5.2 Output port

The eight bit parallel output port **Q0-7** can be.controlled by a transputer family device via the serial link. **QValid** and **QAck** provide a simple two-wire handshake for this port.

A data packet received on the input link is presented on **Q0-7**; the link adaptor then takes **QValid** high to initiate the handshake. After reading data from **Q0-7**, the peripheral device sets **QAck** high. The IMS C011 will then send an acknowledge packet out of the serial link to indicate a completed transaction and set **QValid** low to complete the handshake.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TLdVQvH	Start of link data to **QValid**	11.5			bits	1
TQdVQvH	Data setup	12			ns	2
TQvHQaH	**QAck** setup time from **QValid** high	0			ns	
TQaHQvL	**QAck** high to **QValid** low	1.8			bits	1
TQaHLaV	**QAck** high to Ack on link	0.8		2.5	bits	1,3
TQvLQaL	**QAck** hold after **QValid** low	0			ns	
TQvLQdX	Data hold	11			bits	1,4

Notes

1 Unit of measurement is one link data bit time; at 10 Mbits/s data link speed, one bit time is nominally 100 ns.

2 Where an existing data output bit is re-written with the same level there will be no glitch in the output level.

3 Maximum time assumes there is no data packet already on the link. Maximum time with data on the link is extended by 11 bits.

4 Data output remains valid until overwritten by new data.

Table 5.2 Mode 1 parallel data output

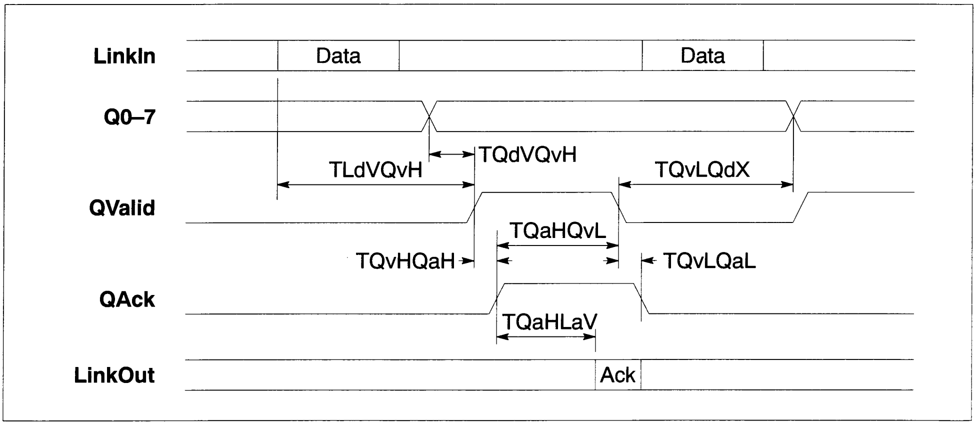


Figure 5.2 IMS C011 Mode 1 parallel data output from link adaptor

6 Mode 2 parallel interface

The IMS C011 provides an interface between a link and a microprocessor style bus. Operation of the link adaptor is controlled through the parallel interface bus lines **D0-7** by reading and writing various registers in the link adaptor. Registers are selected by **RS0-1** and **RnotW**, and the chip enabled with **notCS**.

For convenience of description, the device connected to the parallel side of the link adaptor is presumed to be a microprocessor, although this will not always be the case.

6.1 D0–7

Data is communicated between a microprocessor bus and the link adaptor via the bidirectional bus lines **D0-7**. The bus is high impedance unless the link adaptor chip is selected and the **RnotW** line is high. The bus is used by the microprocessor to access status and data registers.

6.2 notCS

The link adaptor chip is selected when **notCS** is low. Register selectors **RS0-1** and **RnotW** must be valid before **notCS** goes low; **D0-7** must also be valid if writing to the chip (**RnotW** low). Data is read by the link adaptor on the rising edge of **notCS**.

6.3 RnotW

RnotW, in conjunction with **notCS**, selects the link adaptor registers for read or write mode. When **RnotW** is high, the contents of an addressed register appear on the data bus **D0-7**; when **RnotW** is low the data on **D0-7** is written into the addressed register. The state of **RnotW** is latched into the link adaptor by **notCS** going low; it may be changed before **notCS** returns high, within the timing restrictions given.

6.4 RS0–1

One of four registers is selected by **RS0-1**. A register is addressed by setting up **RS0-1** and then taking **notCS** low; the state of **RnotW** when **notCS** goes low determines whether the register will be read or written. The state of **RS0-1** is latched into the link adaptor by **notCS** going low; it may be changed before **notCS** returns high, within the timing restrictions given. The register set comprises a read-only data input register, a write-only data output register and a read/write status register for each.

RS1	RS0	RnotW	Register
0	0	1	Read data
0	0	0	Invalid
0	1	1	Invalid
0	1	0	Write data
1	0	1	Read input status
1	0	0	Write input status
1	1	1	Read output status
1	1	0	Write output status

Table 6.1 IMS C011 Mode 2 register selection

6.4.1 Input Data Register

This register holds the last data packet received from the serial link. It never contains acknowledge packets. It contains valid data only whilst the *data present* flag is set in the input status register. It cannot be assumed to contain valid data after it has been read; a double read may or may not return valid data on the second read. If *data present* is valid on a subsequent read it indicates new data is in the buffer. Writing to this register will have no effect.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TRSVCSL	Register select setup	5			ns	
TCSLRSX	Register select hold	8			ns	
TRWVCSL	Read/write strobe setup	5			ns	
TCSLRWX	Read/write strobe hold	8			ns	
TCSLCSH	Chip select active	60			ns	
TCSHCSL	Delay before re-assertion of chip select	50			ns	

Table 6.2 IMS C011 Mode 2 parallel interface control

Symbol	Parameter	Min	Nom	Max	Units	Notes
TLdVIIH	Start of link data to **InputInt** high			14	bits	1
TCSLIIL	Chip select to **InputInt** low			35	ns	
TCSLDrX	Chip select to bus active	5			ns	
TCSLDrV	Chip select to data valid			50	ns	
TCSHDrZ	Chip select high to bus tristate			38	ns	
TCSHDrX	Data hold after chip select high	5			ns	
TCSHLaV	Chip de-select to start of Ack	0.8		2.5	bits	1,2

Notes

1 Unit of measurement is one link data bit time; at 10 Mbits/s data link speed, one bit time is nominally 100 ns.

2 Maximum time assumes there is no data packet already on the link. Maximum time with data on the link is extended by 11 bits.

Table 6.3 IMS C011 Mode 2 parallel interface read

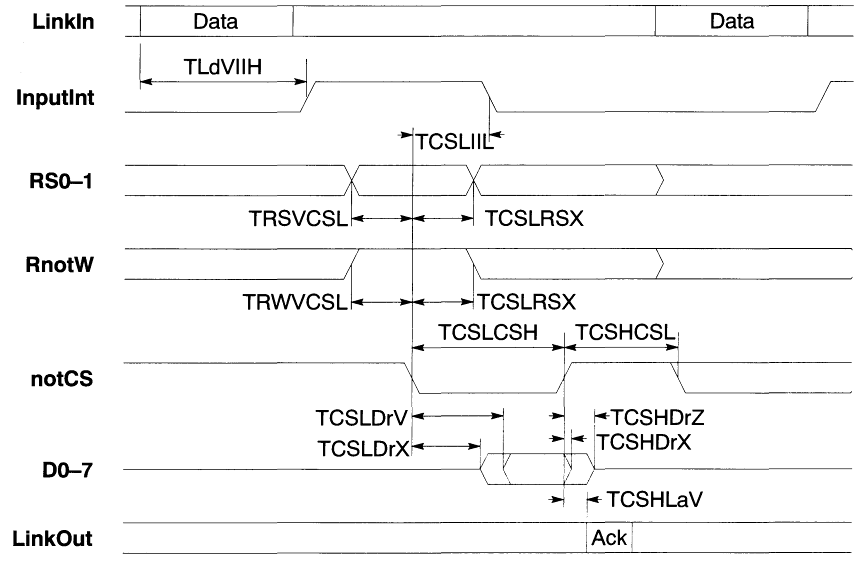


Figure 6.1 IMS C011 Mode 2 read parallel data from link adaptor

Symbol	Parameter	Min	Nom	Max	Units	Notes
TCSHDwV	Data setup	10			ns	
TCSHDwX	Data hold	10			ns	
TCSLOIL	Chip select to **OutputInt** low			35	ns	
TCSHLdV	Chip select high to start of link data	0.8		2.5	bits	1,2
TLaVOIH	Start of link Ack to **OutputInt** high			3.3	bits	1,3
TLdVOIH	Start of link data to **OutputInt** high			13	bits	1,3

Notes

1 Unit of measurement is one link data bit time; at 10 Mbits/s data link speed, one bit time is nominally 100 ns.

2 Maximum time assumes there is no acknowledge packet already on the link. Maximum time with acknowledge on the link is extended by 2 bits.

3 Both data transmission and the returned acknowledge must be completed before **OutputInt** can go high.

Table 6.4 IMS C011 Mode 2 parallel interface write

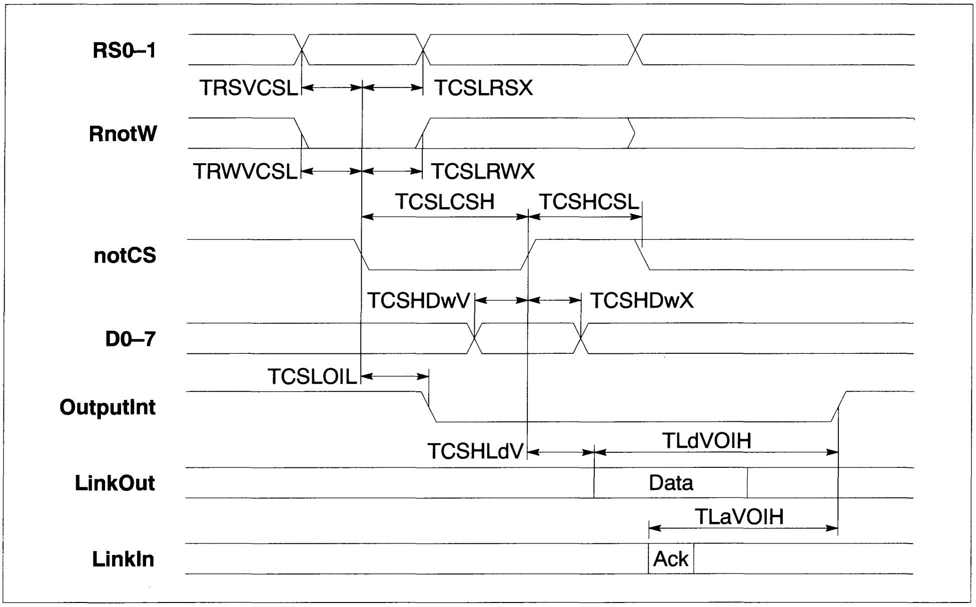


Figure 6.2 IMS C011 Mode 2 write parallel data to link adaptor

6.4.2 Input Status Register

This register contains the *data present* flag and the *interrupt enable* control bit for **InputInt**. The *data present* flag is set to indicate that data in the data input buffer is valid. It is reset low only when the data input buffer is read, or by **Reset**. When writing to this register, the *data present* bit must be written as zero.

The *interrupt enable* bit can be set and reset by writing to the status register with this bit high or low respectively. When the *interrupt enable* and *data present* flags are both high, the **InputInt** output will be high (section 6.5). Resetting *interrupt enable* will take **InputInt** low; setting it again before reading the data input register will set **InputInt** high again. The *interrupt enable* bit can be read to determine its status.

When writing to this register, bits 2-7 must be written as zero; this ensures that they will be zero when the register is read. Failure to write zeroes to these bits may result in undefined data being returned by these bits during a status register read.

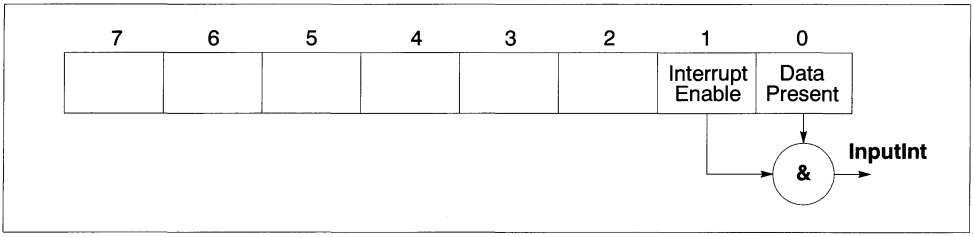


Figure 6.3 IMS C011 Mode 2 input status register

6.4.3 Output Data Register

Data written to this link adaptor register is transmitted out of the serial link as a data packet. Data should only be written to this register when the *output ready* bit in the output status register is high, otherwise data already being transmitted may be corrupted. Reading this register will result in undefined data being read.

6.4.4 Output Status Register

This register contains the *output ready* flag and the *interrupt enable* control bit for **OutputInt**. The *output ready* flag is set to indicate that the data output buffer is empty and a link acknowledge has been received. It is reset low only when data is written to the data output buffer; it is set high by **Reset**. When writing to this register, the *output ready* bit must be written as zero.

The *interrupt enable* bit can be set and reset by writing to the status register with this bit high or low respectively. When the *interrupt enable* and *output ready* flags are both high, the **OutputInt** output will be high (section 6.6). Resetting *interrupt enable* will take **OutputInt** low; setting it again whilst the data output register is empty will set **OutputInt** high again. The *interrupt enable* bit can be read to determine its status.

When writing to this register, bits 2-7 must be written as zero; this ensures that they will be zero when the register is read. Failure to write zeroes to these bits may result in undefined data being returned by these bits during a status register read.

6.5 InputInt

The **InputInt** output is set high to indicate that a data packet has been received from the serial link. It is inhibited from going high when the *interrupt enable* bit in the input status register is low (section 6.4.2). **InputInt** is reset low when data is read from the input data register (section 6.4.1) and by **Reset** (page 422).

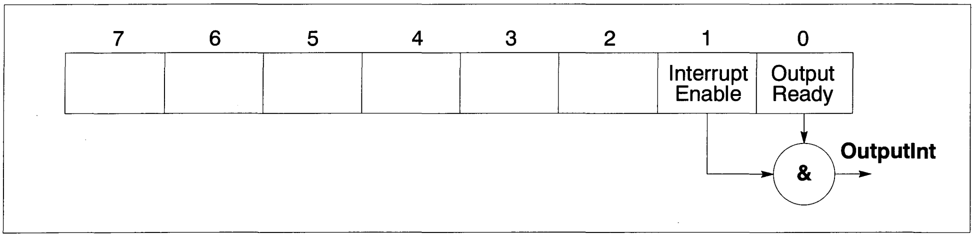


Figure 6.4 IMS C011 Mode 2 output status register

6.6 OutputInt

The **OutputInt** output is set high to indicate that the link is free to receive data from the microprocessor for transmission as a data packet out of the serial link. It is inhibited from going high when the *interrupt enable* bit in the output status register is low (section 6.4.4). **OutputInt** is reset low when data is written to the data output register; it is set low by **Reset** (page 422).

6.7 Data read

A data packet received on the input link sets the *data present* flag in the input status register. If the *interrupt enable* bit in the status register is set, the **InputInt** output pin will be set high. The microprocessor will either respond to the interrupt (if the *interrupt enable* bit is set) or will periodically read the input status register until the *data present* bit is high.

When data is available from the link, the microprocessor reads the data packet from the data input register. This will reset the *data present* flag and cause the link adaptor to transmit an acknowledge packet out of the serial link output. **InputInt** is automatically reset by reading the data input register; it is not necessary to read or write the input status register.

6.8 Data write

When the data output buffer is empty and a link acknowledge has been received the *output ready* flag in the output status register is set high. If the *interrupt enable* bit in the status register is set, the **OutputInt** output pin will also be set high. The microprocessor will either respond to the interrupt (if the *interrupt enable* bit is set) or will periodically read the output status register until the *output ready* bit is high.

When the *output ready* flag is high, the microprocessor can write data to the data output buffer. This will result in the link adaptor resetting the *output ready* flag and commencing transmission of the data packet out of the serial link. The *output ready* status bit will remain low until the data byte transmission has been completed and an acknowledge packet is received by the input link. This will set the *output ready* flag high; if the *interrupt enable* bit is set, **OutputInt** will also be set high.

7 Electrical specifications

7.1 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
II	Input current		±25	mA	4
tOSC	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		600	mW	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 7.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1,2
CL	Load capacitance on any pin		60	pF	
TA	Operating temperature range	0	70	°C	3

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Air flow rate 400 linear ft/min transverse air flow.

Table 7.2 Operating conditions

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VIH	High level input voltage	2.0	VDD+0.5	V	1, 2
VIL	Low level input voltage	–0.5	0.8	V	1, 2
II	Input current @ GND<VI<VDD		±10	µA	1, 2, 3
			±200	µA	1, 2, 4
VOH	Output high voltage @ IOH=2mA	VDD–1		V	1, 2
VOL	Output low voltage @ IOL=4mA		0.4	V	1, 2
IOZ	Tristate output current @ GND<V0<VDD		±10	µA	1, 2
PD	Power dissipation		120	mW	2, 5
CIN	Input capacitance @ f=1MHz		7	pF	6
COZ	Output capacitance @ f=1MHz		10	pF	6

Notes

1 All voltages are with respect to **GND**.

2 Parameters for IMS C011 measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C.

3 For inputs other than those in Note 4.

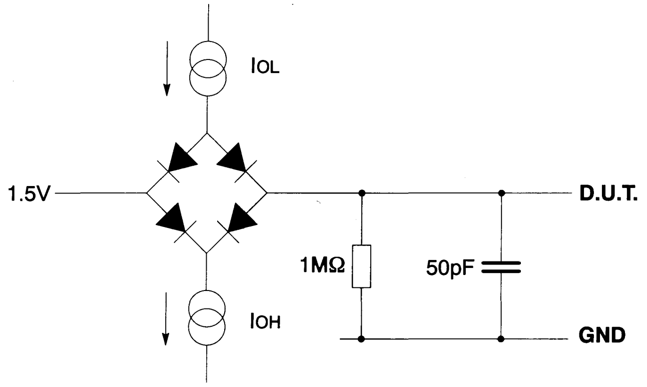
4 For pins 2, 3, 5, 6, 7, 9, 11, 13, 15, 16, 25.

5 Power dissipation varies with output loading.

6 This parameter is sampled and not 100% tested.

Table 7.3 DC characteristics

7.2 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 7.1 Load circuit for AC measurements

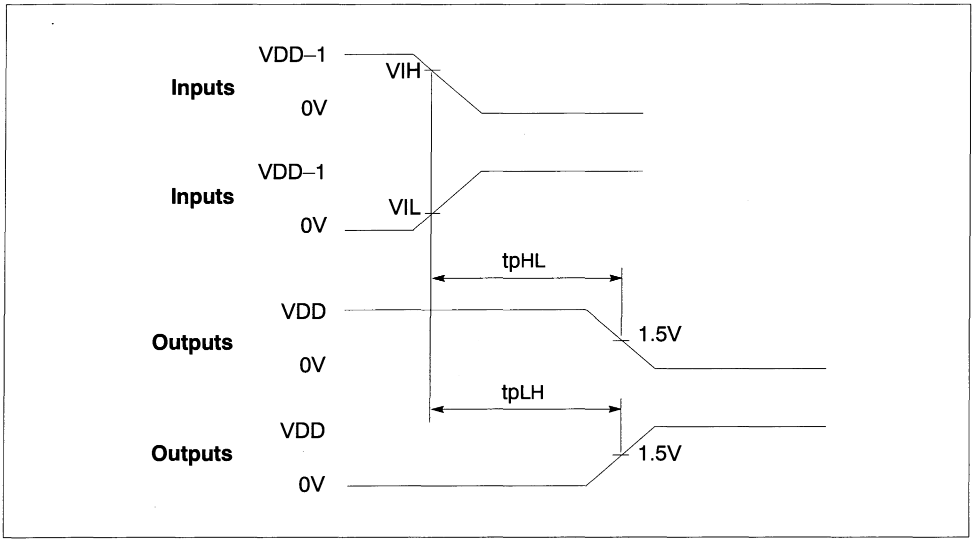


Figure 7.2 AC measurements timing waveforms

7.3 AC timing characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3
TQr	Output rising edges		25	ns	1, 4
TQf	Output falling edges		15	ns	1, 4
TCSLaHZ	Chip select high to tristate		25	ns	
TCSLaLZ	Chip select low to tristate		25	ns	

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 These parameters are not tested.

4 This parameter is sampled and is not 100% tested.

Table 7.4 Input and output edges

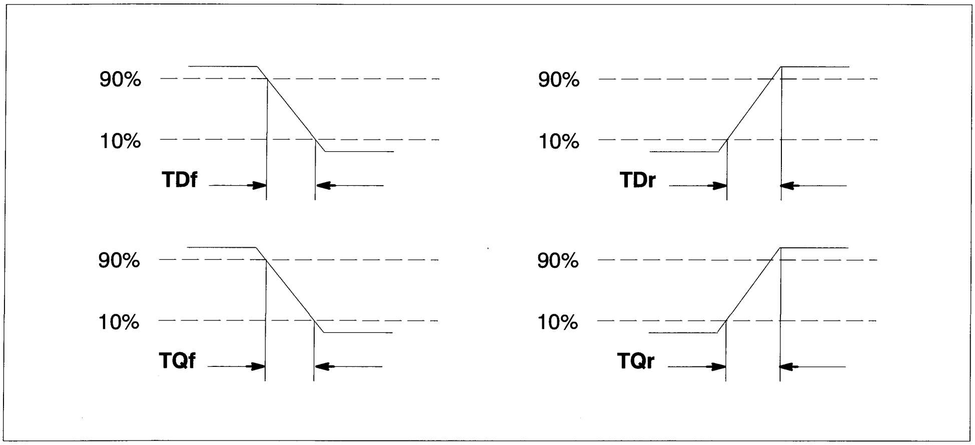


Figure 7.3 IMS C011 input and output edge timing

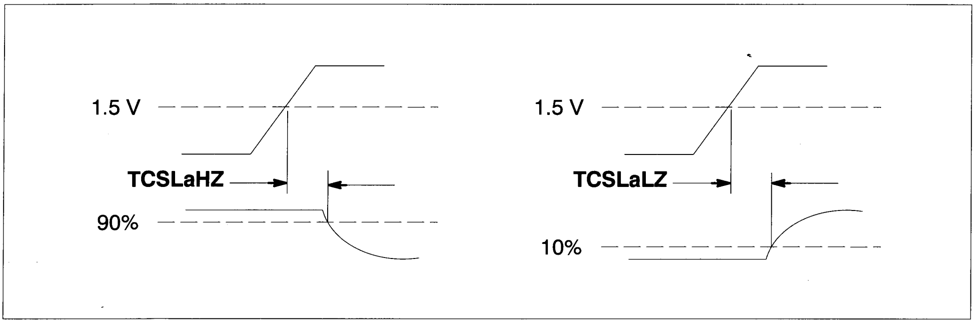


Figure 7.4 IMS C011 tristate timing relative to **notCS**

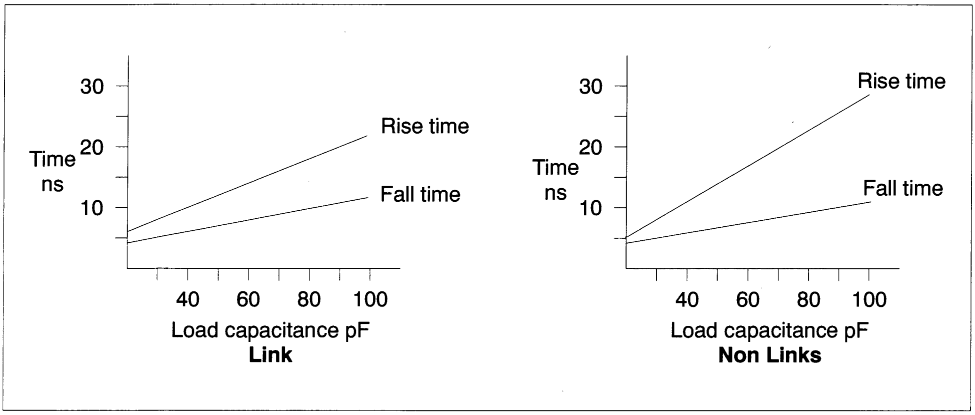


Figure 7.5 Typical rise/fall times

7.4 Power rating

Internal power dissipation (P_{INT}) of transputer and peripheral chips depends on **VDD**, as shown in figure 7.6. P_{INT} is substantially independent of temperature.

Total power dissipation (P_D) of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

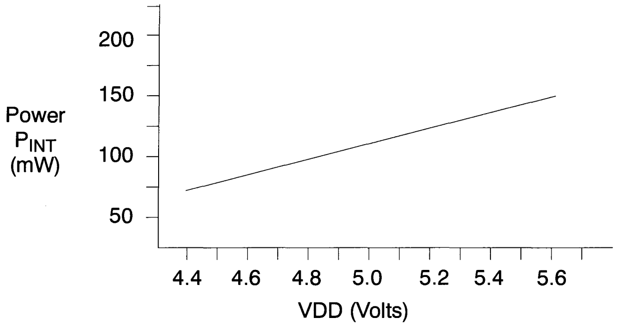


Figure 7.6 IMS C011 internal power dissipation vs VDD

8 Package pinouts

8.1 28 pin DIL package and 28 pin SOJ package pinout

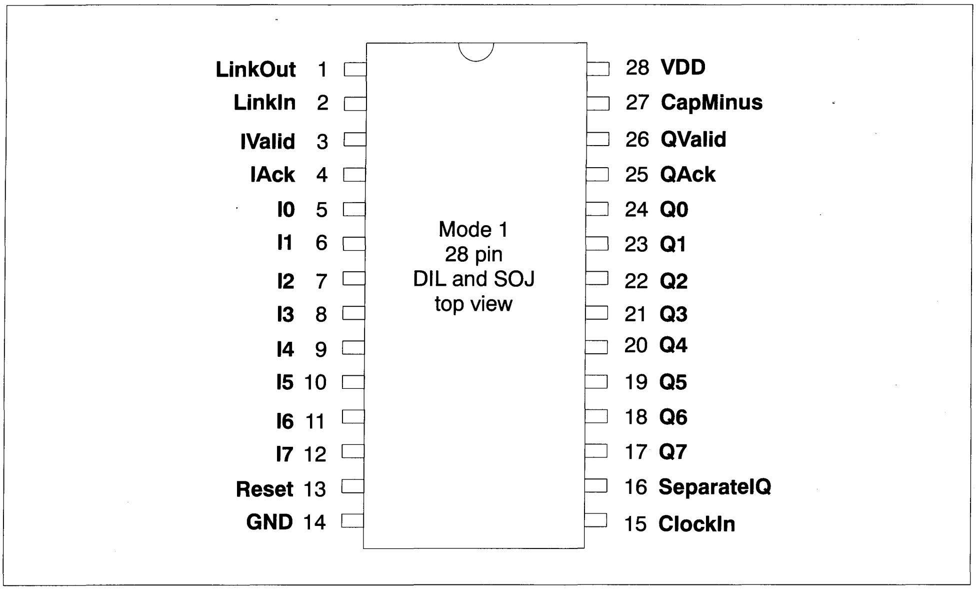


Figure 8.1 IMS C011 **Mode 1** 28 pin DIL and SOJ package pinout

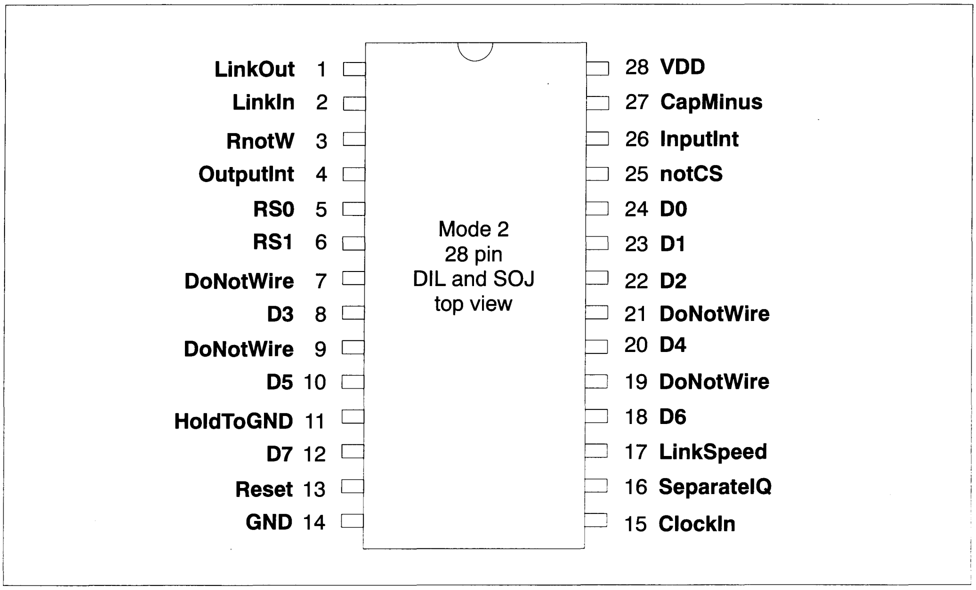


Figure 8.2 IMS C011 **Mode 2** 28 pin DIL and SOJ package pinout

9 Ordering

This section indicates the designation of package selections for the IMS C011. Speed of **ClockIn** is 5 MHz for all parts.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	Package
IMS C011-P20S	28 pin plastic dual-in-line
IMS C011-E20S	28 pin SOJ

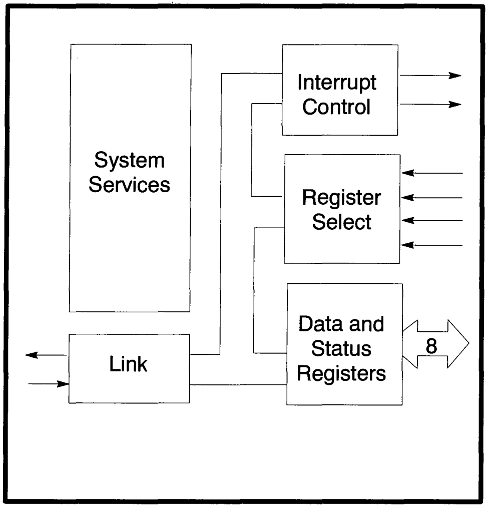
Table 9.1 IMS C011 ordering details

Military versions to MIL-STD-883 are available, see *'The Military and Space Transputer Databook'* for details.

IMS C012 link adaptor

inmos®

Engineering Data



FEATURES

Standard INMOS link protocol
10 or 20 Mbits/sec operating speed
Communicates with INMOS transputers
Converts between serial link and parallel bus
Tristate bidirectional bus interface
Memory mapped registers
Interrupt capability
Single +5V ±5% power supply
TTL and CMOS compatibility
120mW power dissipation
Standard 24 pin 0.3" plastic package

APPLICATIONS

Connecting microprocessors to transputers
High speed links between microprocessors
Inter-family microprocessor interfacing

SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

April 1992
42 1413 03

1 Introduction

The INMOS communication link is a high speed system interconnect which provides full duplex communication between members of the INMOS transputer family, according to the INMOS serial link protocol. The IMS C012, a member of this family, provides for full duplex transputer link communication with standard microprocessor and sub-system architectures, by converting bi-directional serial link data into parallel data streams.

All INMOS products which use communication links, regardless of device type, support a standard communications frequency of 10 Mbits/sec; most products also support 20 Mbits/sec. Products of different type or performance can, therefore, be interconnected directly and future systems will be able to communicate directly with those of today. The IMS C012 link will run at either the standard speed of 10 Mbits/sec or at the higher speed of 20 Mbits/sec. Data reception is asynchronous, allowing communication to be independent of clock phase.

The IMS C012 provides an interface between an INMOS serial link and a microprocessor system bus. Status and data registers for both input and output ports can be accessed across the byte-wide bi-directional interface. Two interrupt outputs are provided, one to indicate input data available and one for output buffer empty.

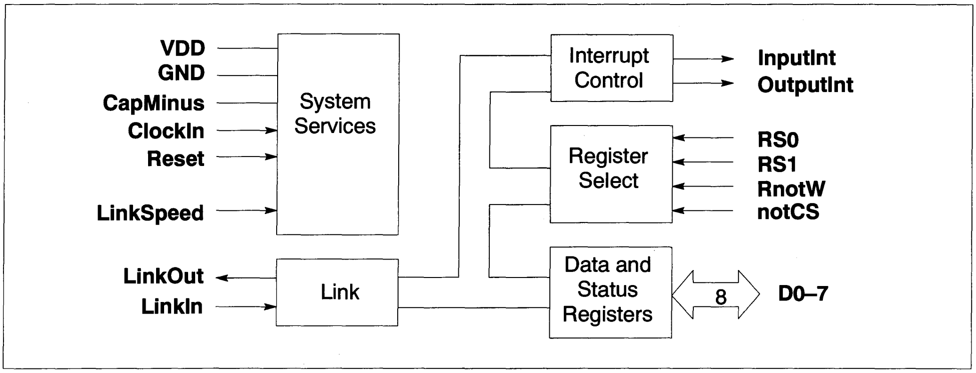


Figure 1.1 IMS C012 block diagram

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high. Pinout details for various packages are given on page 460.

Pin	In/Out	Function
VDD, GND		Power supply and return
CapMinus		External capacitor for internal clock power supply
ClockIn	in	Input clock
Reset	in	System reset
LinkIn	in	Serial data input channel
LinkOut	out	Serial data output channel

Table 2.1 Services and link

Pin	In/Out	Function
D0-7	in/out	Bi-directional data bus
notCS	in	Chip select
RS0-1	in	Register select
RnotW	in	Read/write control signal
InputInt	out	Interrupt on link receive buffer full
OutputInt	out	Interrupt on link transmit buffer empty
LinkSpeed	in	Select link speed as 10 or 20 Mbits/sec
HoldToGND		Must be connected to **GND**

Table 2.2 Parallel interface

3 System services

System services include all the necessary logic to start up and maintain the IMS C012.

3.1 Power

Power is supplied to the device via the **VDD** and **GND** pins. The supply must be decoupled close to the chip by at least one 100 nF low inductance (e.g. ceramic) capacitor between **VDD** and **GND**. Four layer boards are recommended; if two layer boards are used, extra care should be taken in decoupling.

AC noise between **VDD** and **GND** must be kept below 200 mV peak to peak at all frequencies above 100 KHz. AC noise between **VDD** and the ground reference of load capacitances must be kept below 200 mV peak to peak at all frequencies above 30 MHz. Input voltages must not exceed specification with respect to **VDD** and **GND**, even during power-up and power-down ramping, otherwise *latchup* can occur. CMOS devices can be permanently damaged by excessive periods of latchup.

3.2 CapMinus

The internally derived power supply for internal clocks requires an external low leakage, low inductance 1μF capacitor to be connected between **VDD** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 Ohms between 100 KHz and 10 MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50 mm. The positive connection of the capacitor must be connected directly to **VDD**. Connections must not otherwise touch power supplies or other noise sources.

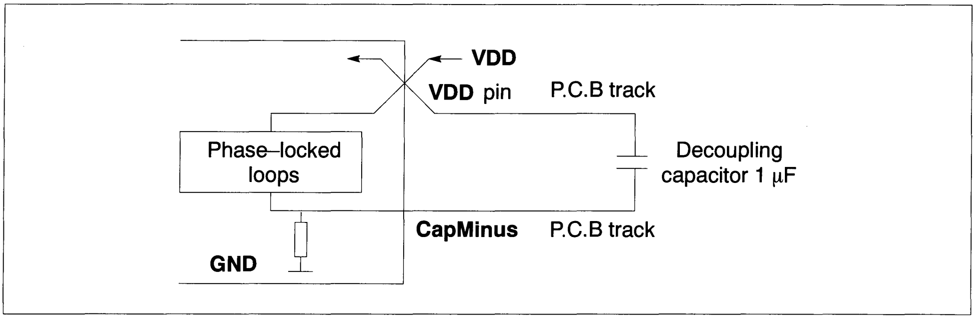


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

Transputer family components use a standard clock frequency, supplied by the user on the **ClockIn** input. The nominal frequency of this clock for all transputer family components is 5 MHz, regardless of device type, transputer word length or processor cycle time. High frequency internal clocks are derived from **ClockIn**, simplifying system design and avoiding problems of distributing high speed clocks externally.

A number of transputer family devices may be connected to a common clock, or may have individual clocks providing each one meets the specified stability criteria. In a multi-clock system the relative phasing of **ClockIn** clocks is not important, due to the asynchronous nature of the links. Mark/space ratio is unimportant provided the specified limits of **ClockIn** pulse widths are met.

Oscillator stability is important. **ClockIn** must be derived from a crystal oscillator; RC oscillators are not sufficiently stable. **ClockIn** must not be distributed through a long chain of buffers. Clock edges must be monotonic and remain within the specified voltage and time limits.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TDCLDCH	**ClockIn** pulse width low	40			ns	1
TDCHDCL	**ClockIn** pulse width high	40			ns	1
TDCLDCL	**ClockIn** period		200	400	ns	1,2,4
TDCerror	**ClockIn** timing error			±0.5	ns	1,3
TDC1DC2	Difference in **ClockIn** for 2 linked devices			400	ppm	1,4
TDCr	**ClockIn** rise time			10	ns	1,5
TDCf	**ClockIn** fall time			8	ns	1,5

Notes

1. Guaranteed, but not tested.
2. Measured between corresponding points on consecutive falling edges.
3. Variation of individual falling edges from their nominal times.
4. This value allows the use of 200ppm crystal oscillators for two devices connected together by a link.
5. Clock transitions must be monotonic within the range **VIH** to **VIL** (table 6.3).

Table 3.1 Input clock

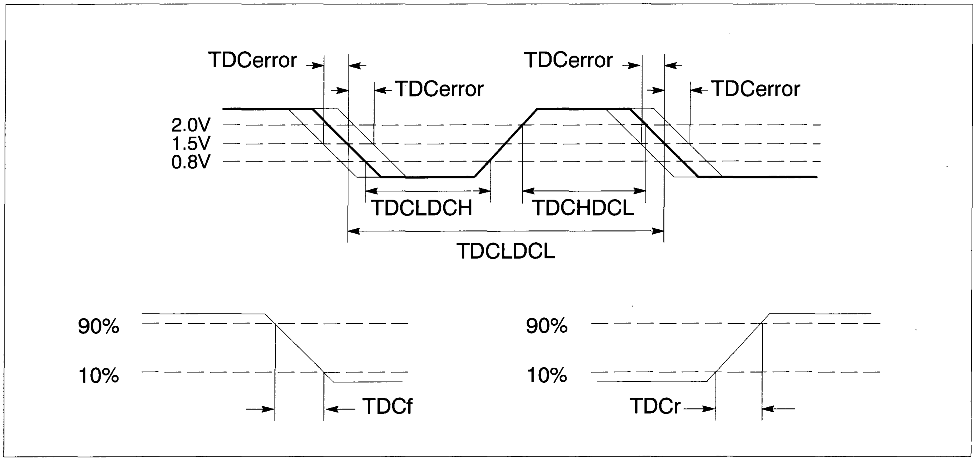


Figure 3.2 ClockIn timing

3.4 Reset

The **Reset** pin can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. All unputs, with the exception of **ClockIn**, **SeparateIQ** and **LinkSpeed**, must be held in their inactive state during reset.

Reset initialises the IMS C012 to the following state: **LinkOut** is held low; the interrupt outputs **InputInt** and **OutputInt** are held low; interrupts are disabled; **D0-7** are high impedance.

Symbol	Parameter	Min	Nom	Max	Units	Notes
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TRLCSL	**Reset** low before chip select low	0			ns	

Notes

1 Full periods of **ClockIn TDCLDCL** required.

2 At power-on reset.

Table 3.2 Reset

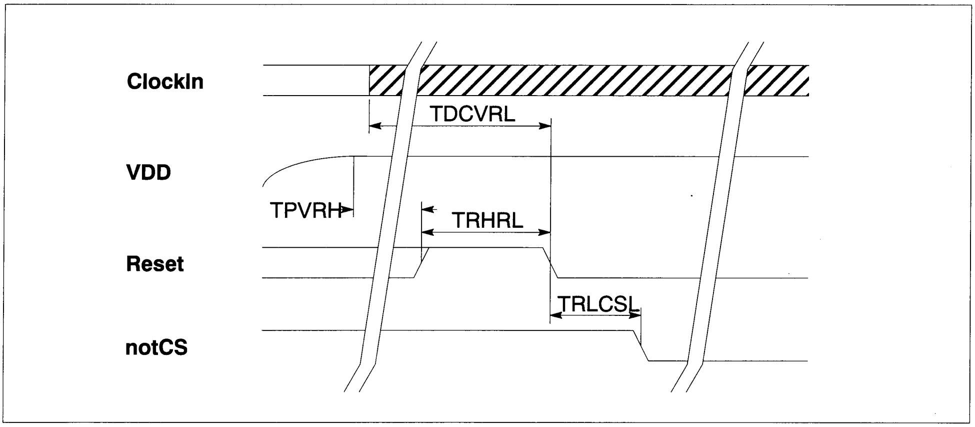


Figure 3.3 **Reset** timing

4 Links

INMOS bi-directional serial links provide synchronized communication between INMOS products and with the outside world. Each link comprises an input channel and output channel. A link between two devices is implemented by connecting a link interface on one device to a link interface on the other device. Every byte of data sent on a link is acknowledged on the input of the same link, thus each signal line carries both data and control information.

The quiescent state of a link output is low. Each data byte is transmitted as a high start bit followed by a one bit followed by eight data bits followed by a low stop bit. The least significant bit of data is transmitted first. After transmitting a data byte the sender waits for the acknowledge, which consists of a high start bit followed by a zero bit. The acknowledge signifies both that a process was able to receive the acknowledged data byte and that the receiving link is able to receive another byte.

Links are not synchronised with **ClockIn** and are insensitive to its phase. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

Links are TTL compatible and intended to be used in electrically quiet environments, between devices on a single printed circuit board or between two boards via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure that the reflection returns before the next data bit is sent.

Buffers may be used for very long transmissions. If so, their overall propagation delay should be stable within the skew tolerance of the link, although the absolute value of the delay is immaterial.

The IMS C012 link supports the standard INMOS communication speed of 10 Mbits/sec. In addition it can be used at 20 Mbits/sec. Link speed can be selected by **LinkSpeed**; when the **LinkSpeed** pin is low, the link operates at the standard 10 Mbits/sec; when high it operates at 20 Mbits/sec.

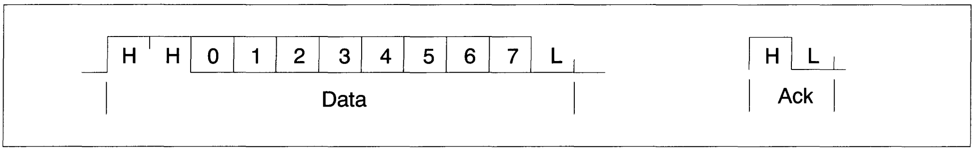


Figure 4.1 IMS C012 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	1
TJQf	**LinkOut** fall time				10	ns	1
TJDr	**LinkIn** rise time				20	ns	3
TJDf	**LinkIn** fall time				20	ns	3
TJQJD	Buffered edge delay		0			ns	
TJBskew	Variation in TJQJD	20 Mbits/s			3	ns	2
		10 Mbits/s			10	ns	2
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	1
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100Ω transmission line			56		ohms	

Notes

1 This parameter is sampled but is not 100% tested.

2 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.

3 This parameter is not tested.

Table 4.1 Link

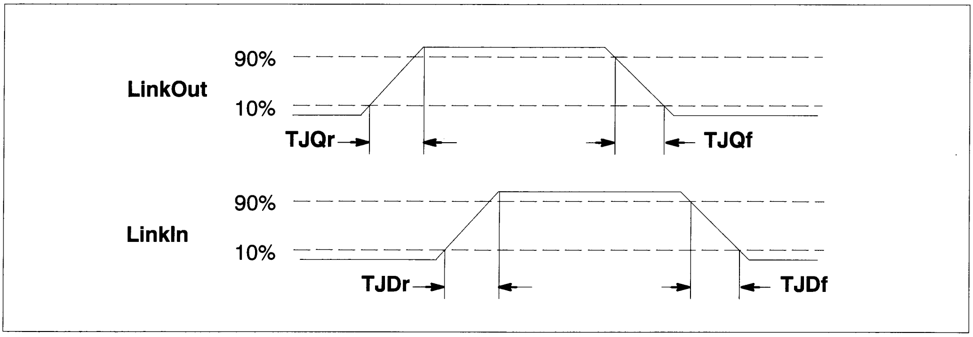


Figure 4.2 IMS C012 link timing

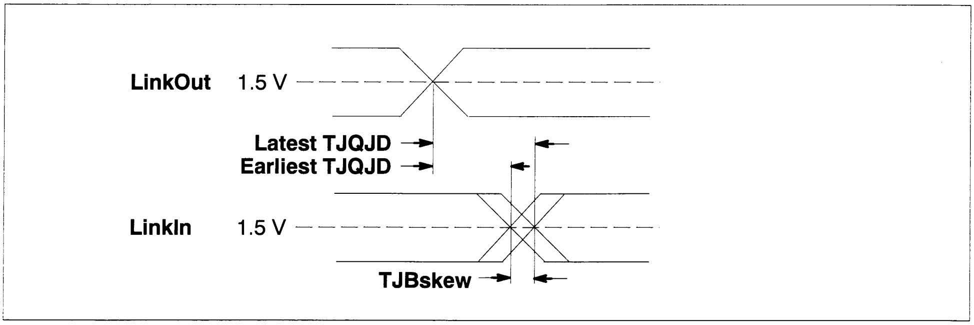


Figure 4.3 IMS C012 buffered link timing

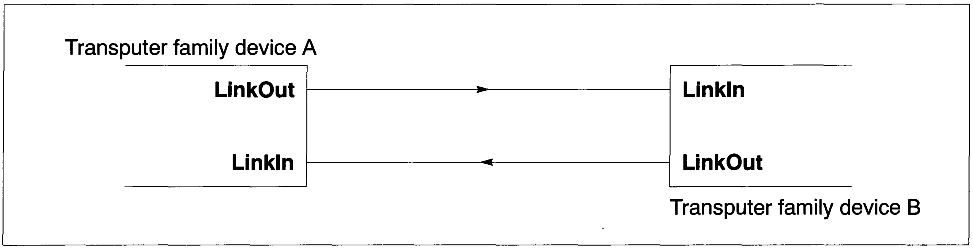


Figure 4.4 Links directly connected

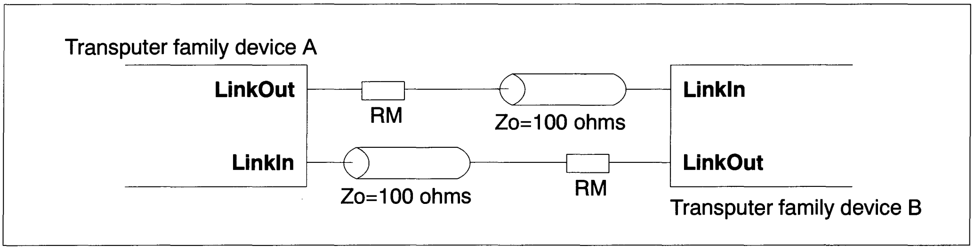


Figure 4.5 Links connected by transmission line

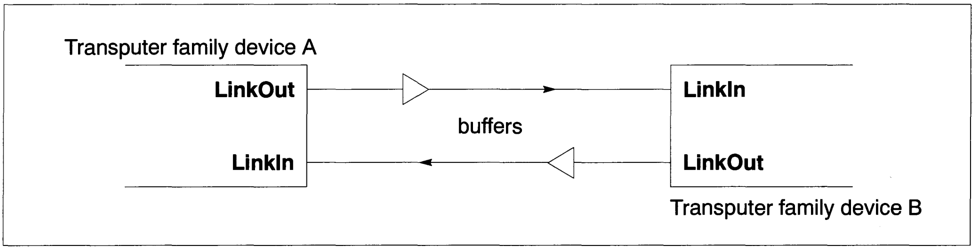


Figure 4.6 Links connected by buffers

5 Parallel interface

The IMS C012 provides an interface between a link and a microprocessor style bus. Operation of the link adaptor is controlled through the parallel interface bus lines **D0-7** by reading and writing various registers in the link adaptor. Registers are selected by **RS0-1** and **RnotW**, and the chip enabled with **notCS**.

For convenience of description, the device connected to the parallel side of the link adaptor is presumed to be a microprocessor, although this will not always be the case.

5.1 D0–7

Data is communicated between a microprocessor bus and the link adaptor via the bidirectional bus lines **D0-7**. The bus is high impedance unless the link adaptor chip is selected and the **RnotW** line is high. The bus is used by the microprocessor to access status and data registers.

5.2 notCS

The link adaptor chip is selected when **notCS** is low. Register selectors **RS0-1** and **RnotW** must be valid before **notCS** goes low; **D0-7** must also be valid if writing to the chip (**RnotW** low). Data is read by the link adaptor on the rising edge of **notCS**.

5.3 RnotW

RnotW, in conjunction with **notCS**, selects the link adaptor registers for read or write mode. When **RnotW** is high, the contents of an addressed register appear on the data bus **D0-7**; when **RnotW** is low the data on **D0-7** is written into the addressed register. The state of **RnotW** is latched into the link adaptor by **notCS** going low; it may be changed before **notCS** returns high, within the timing restrictions given.

5.4 RS0–1

One of four registers is selected by **RS0-1**. A register is addressed by setting up **RS0-1** and then taking **notCS** low; the state of **RnotW** when **notCS** goes low determines whether the register will be read or written. The state of **RS0-1** is latched into the link adaptor by **notCS** going low; it may be changed before **notCS** returns high, within the timing restrictions given. The register set comprises a read-only data input register, a write-only data output register and a read/write status register for each.

RS1	RS0	RnotW	Register
0	0	1	Read data
0	0	0	Invalid
0	1	1	Invalid
0	1	0	Write data
1	0	1	Read input status
1	0	0	Write input status
1	1	1	Read output status
1	1	0	Write output status

Table 5.1 IMS C012 register selection

5.4.1 Input Data Register

This register holds the last data packet received from the serial link. It never contains acknowledge packets. It contains valid data only whilst the *data present* flag is set in the input status register. It cannot be assumed to contain valid data after it has been read; a double read may or may not return valid data on the second read. If *data present* is valid on a subsequent read it indicates new data is in the buffer. Writing to this register will have no effect.

Symbol	Parameter	Min	Nom	Max	Units	Note
TRSVCSL	Register select setup	5			ns	
TCSLRSX	Register select hold	8			ns	
TRWVCSL	Read/write strobe setup	5			ns	
TCSLRWX	Read/write strobe hold	8			ns	
TCSLCSH	Chip select active	60			ns	
TCSHCSL	Delay before re-assertion of chip select	50			ns	

Table 5.2 IMS C012 parallel interface control

Symbol	Parameter	Min	Nom	Max	Units	Note
TLdVIIH	Start of link data to **InputInt** high			14	bits	1
TCSLIIL	Chip select to **InputInt** low			35	ns	
TCSLDrX	Chip select to bus active	5			ns	
TCSLDrV	Chip select to data valid			50	ns	
TCSHDrZ	Chip select high to bus tristate			38	ns	
TCSHDrX	Data hold after chip select high	5			ns	
TCSHLaV	Chip de-select to start of Ack	0.8		2.5	bits	1,2

Notes

1 Unit of measurement is one link data bit time; at 10 Mbits/s data link speed, one bit time is nominally 100 ns.

2 Maximum time assumes there is no acknowledge packet already on the link. Maximum time with acknowledge on the link is extended by 11 bits.

Table 5.3 IMS C012 parallel interface read

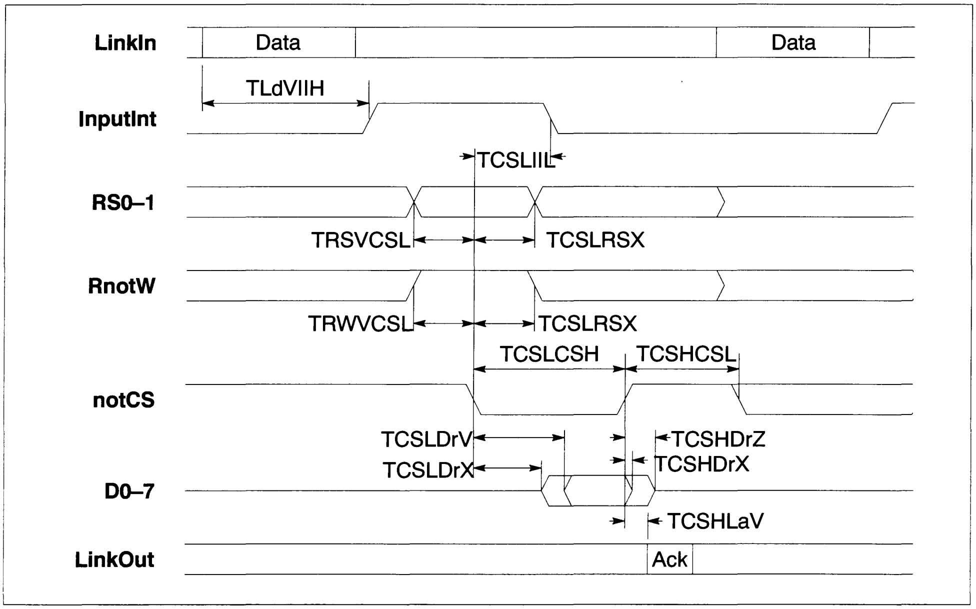


Figure 5.1 IMS C012 read parallel data from link adaptor

Symbol	Parameter	Min	Nom	Max	Units	Note
TCSHDwV	Data setup	10			ns	
TCSHDwX	Data hold	10			ns	
TCSLOIL	Chip select to **OutputInt** low			35	ns	
TCSHLdV	Chip select high to start of link data	0.8		2.5	bits	1,2
TLaVOIH	Start of link Ack to **OutputInt** high			3.3	bits	1,3
TLdVOIH	Start of link data to **OutputInt** high			13	bits	1,3

Notes

1 Unit of measurement is one link data bit time; at 10 Mbits/s data link speed, one bit time is nominally 100 ns.

2 Maximum time assumes there is no acknowledge packet already on the link. Maximum time with acknowledge on the link is extended by 2 bits.

3 Both data transmission and the returned acknowledge must be completed before **OutputInt** can go high.

Table 5.4 IMS C012 parallel interface write

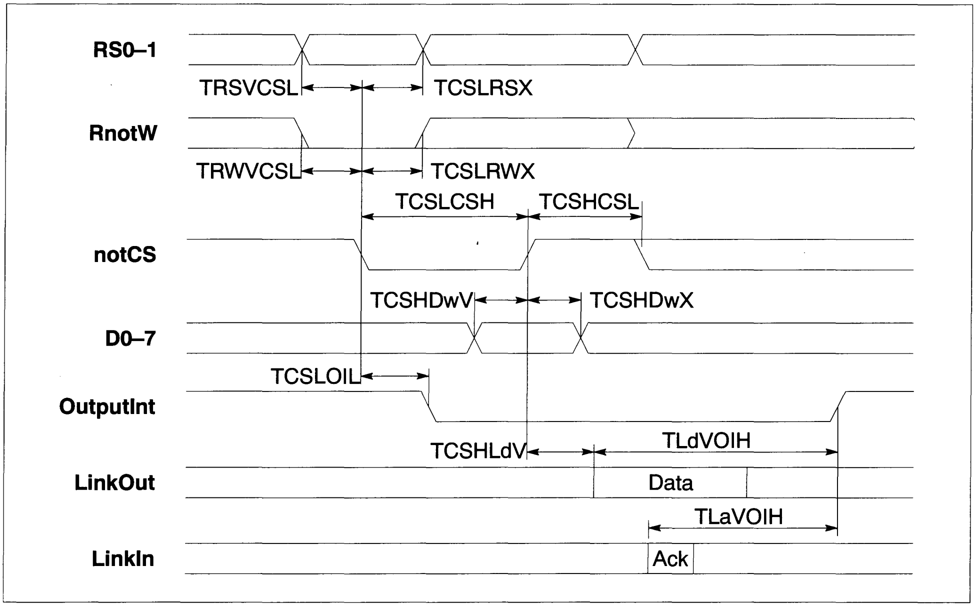


Figure 5.2 IMS C012 write parallel data to link adaptor

5.4.2 Input Status Register

This register contains the *data present* flag and the *interrupt enable* control bit for **InputInt**. The *data present* flag is set to indicate that data in the data input buffer is valid. It is reset low only when the data input buffer is read, or by **Reset**. When writing to this register, the *data present* bit must be written as zero.

The *interrupt enable* bit can be set and reset by writing to the status register with this bit high or low respectively. When the *interrupt enable* and *data present* flags are both high, the **InputInt** output will be high (section 5.5). Resetting *interrupt enable* will take **InputInt** low; setting it again before reading the data input register will set **InputInt** high again. The *interrupt enable* bit can be read to determine its status.

When writing to this register, bits 2-7 must be written as zero; this ensures that they will be zero when the register is read. Failure to write zeroes to these bits may result in undefined data being returned by these bits during a status register read.

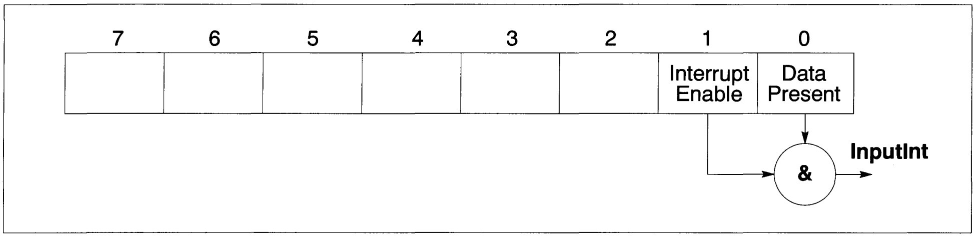


Figure 5.3 IMS C012 input status register

5.4.3 Output Data Register

Data written to this link adaptor register is transmitted out of the serial link as a data packet. Data should only be written to this register when the *output ready* bit in the output status register is high, otherwise data already being transmitted may be corrupted. Reading this register will result in undefined data being read.

5.4.4 Output Status Register

This register contains the *output ready* flag and the *interrupt enable* control bit for **OutputInt**. The *output ready* flag is set to indicate that the data output buffer is empty. It is reset low only when data is written to the data output buffer; it is set high by **Reset**. When writing to this register, the *output ready* bit must be written as zero.

The *interrupt enable* bit can be set and reset by writing to the status register with this bit high or low respectively. When the *interrupt enable* and *output ready* flags are both high, the **OutputInt** output will be high (section 5.6). Resetting *interrupt enable* will take **OutputInt** low; setting it again whilst the data output register is empty will set **OutputInt** high again. The *interrupt enable* bit can be read to determine its status.

When writing to this register, bits 2-7 must be written as zero; this ensures that they will be zero when the register is read. Failure to write zeroes to these bits may result in undefined data being returned by these bits during a status register read.

5.5 InputInt

The **InputInt** output is set high to indicate that a data packet has been received from the serial link. It is inhibited from going high when the *interrupt enable* bit in the input status register is low (section 5.4.2). **InputInt** is reset low when data is read from the input data register (section 5.4.1) and by **Reset** (page 446).

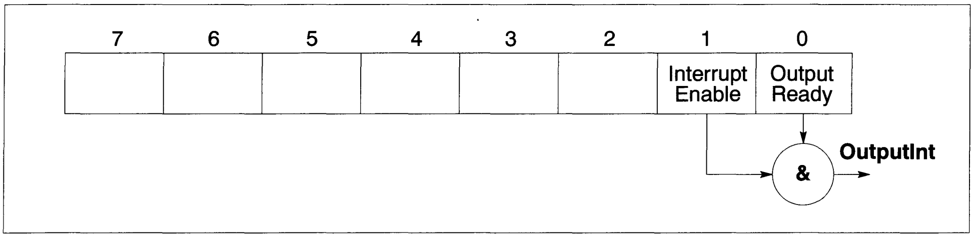


Figure 5.4 IMS C012 output status register

5.6 OutputInt

The **OutputInt** output is set high to indicate that the link is free to receive data from the microprocessor for transmission as a data packet out of the serial link. It is inhibited from going high when the *interrupt enable* bit in the output status register is low (section 5.4.4). **OutputInt** is reset low when data is written to the data output register; it is set low by **Reset** (page 446).

5.7 Data read

A data packet received on the input link sets the *data present* flag in the input status register. If the *interrupt enable* bit in the status register is set, the **InputInt** output pin will be set high. The microprocessor will either respond to the interrupt (if the *interrupt enable* bit is set) or will periodically read the input status register until the *data present* bit is high.

When data is available from the link, the microprocessor reads the data packet from the data input register. This will reset the *data present* flag and cause the link adaptor to transmit an acknowledge packet out of the serial link output. **InputInt** is automatically reset by reading the data input register; it is not necessary to read or write the input status register.

5.8 Data write

When the data output buffer is empty the *output ready* flag in the output status register is set high. If the *interrupt enable* bit in the status register is set, the **OutputInt** output pin will also be set high. The microprocessor will either respond to the interrupt (if the *interrupt enable* bit is set) or will periodically read the output status register until the *output ready* bit is high.

When the *output ready* flag is high, the microprocessor can write data to the data output buffer. This will result in the link adaptor resetting the *output ready* flag and commencing transmission of the data packet out of the serial link. The *output ready* status bit will remain low until the data byte transmission has been completed and an acknowledge packet is received by the input link. This will set the *output ready* flag high; if the *interrupt enable* bit is set, **OutputInt** will also be set high.

6 Electrical specifications

6.1 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
VI, VO	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
II	Input current		±25	mA	4
tOSC	Output short circuit time (one pin)		1	s	2
TS	Storage temperature	–65	150	°C	2
TA	Ambient temperature under bias	–55	125	°C	2
PDmax	Maximum allowable dissipation		600	mW	

Notes

1 All voltages are with respect to **GND**.

2 This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operating sections of this specification is not implied. Stresses greater than those listed may cause permanent damage to the device. Exposure to absolute maximum rating conditions for extended periods may affect reliability.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltage higher than the absolute maximum rated voltages to this high impedance circuit. Unused inputs should be tied to an appropriate logic level such as **VDD** or **GND**.

4 The input current applies to any input or output pin and applies when the voltage on the pin is between **GND** and **VDD**.

Table 6.1 Absolute maximum ratings

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	4.75	5.25	V	1
VI, VO	Input or output voltage	0	VDD	V	1,2
CL	Load capacitance on any pin		60	pF	
TA	Operating temperature range	0	70	°C	3

Notes

1 All voltages are with respect to **GND**.

2 Excursions beyond the supplies are permitted but not recommended; see DC characteristics.

3 Air flow rate 400 linear ft/min transverse air flow.

Table 6.2 Operating conditions

Symbol	Parameter		Min	Max	Units	Notes
V_{IH}	High level input voltage		2.0	VDD+0.5	V	1,2
V_{IL}	Low level input voltage		–0.5	0.8	V	1,2
I_{I}	Input current	@ GND<VI<VDD		±10	μA	1,2,5
				±200	μA	1,2,6
V_{OH}	Output high voltage	@ IOH=2mA	VDD–1		V	1,2
V_{OL}	Output low voltage	@ IOL=4mA		0.4	V	1,2
I_{OZ}	Tristate output current	@ GND<V0<VDD		±10	μA	1,2
P_{D}	Power dissipation			120	mW	2,3
C_{IN}	Input capacitance	@ f=1MHz		7	pF	4
C_{OZ}	Output capacitance	@ f=1MHz		10	pF	4

Notes

1 All voltages are with respect to **GND**.

2 Parameters for IMS C012-S measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C.

3 Power dissipation varies with output loading.

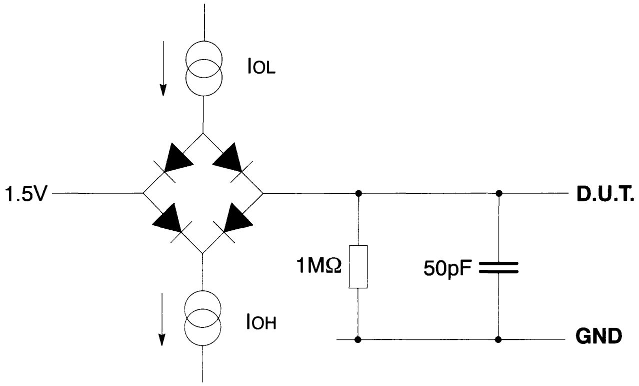
4 This parameter is sampled and not 100% tested.

5 For inputs other than those in Note 6.

6 For pins 2, 3, 5, 6, 9, 11, 13, 14, 21.

Table 6.3 DC characteristics

6.2 Equivalent circuits



Note: This circuit represents the device sinking IOL and sourcing IOH with a 50pF capacitive load.

Figure 6.1 Load circuit for AC measurements

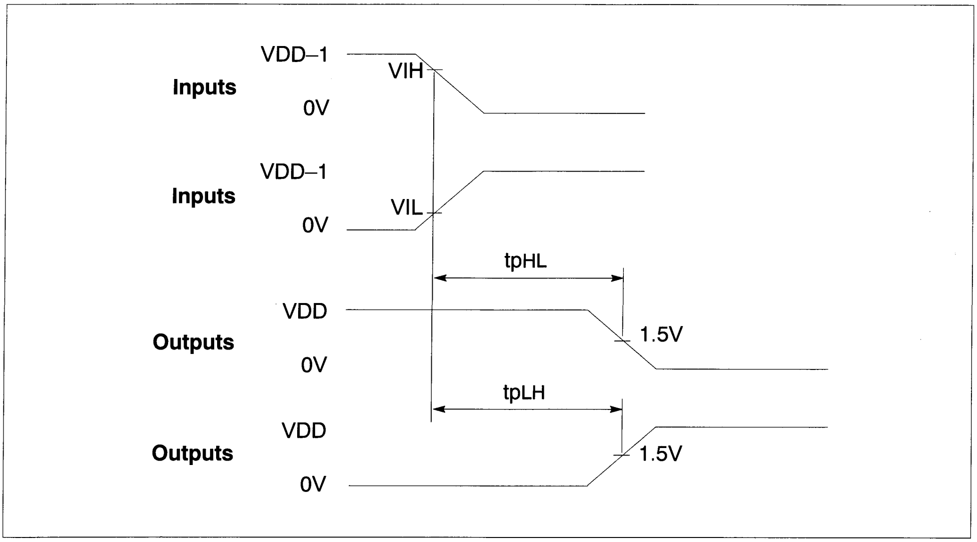


Figure 6.2 AC measurements timing waveforms

6.3 AC timing characteristics

Symbol	Parameter	Min	Max	Units	Notes
TDr	Input rising edges	2	20	ns	1, 2, 3
TDf	Input falling edges	2	20	ns	1, 2, 3
TQr	Output rising edges		25	ns	1, 4
TQf	Output falling edges		15	ns	1, 4
TCSLaHZ	Chip select high to tristate		25	ns	
TCSLaLZ	Chip select low to tristate		25	ns	

Notes

1 Non-link pins; see section on links.

2 All inputs except **ClockIn**; see section on **ClockIn**.

3 This parameter is not tested.

4 This parameter is sampled and is not 100% tested.

Table 6.4 Input and output edges

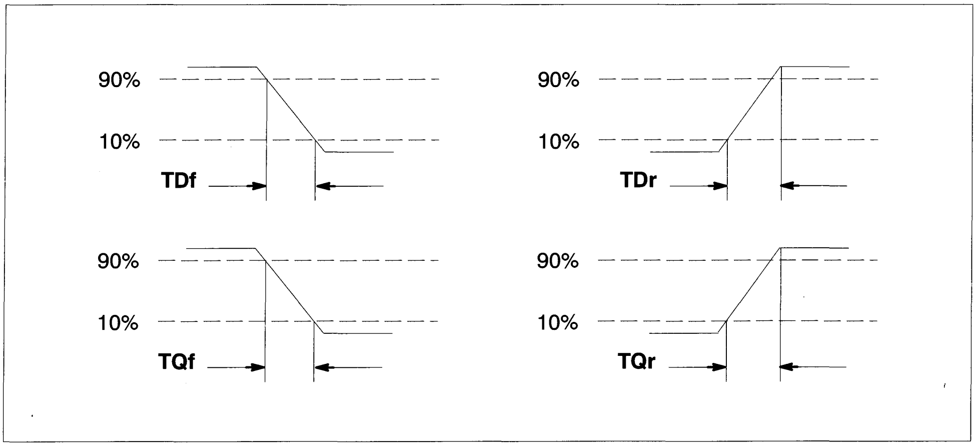


Figure 6.3 IMS C012 input and output edge timing

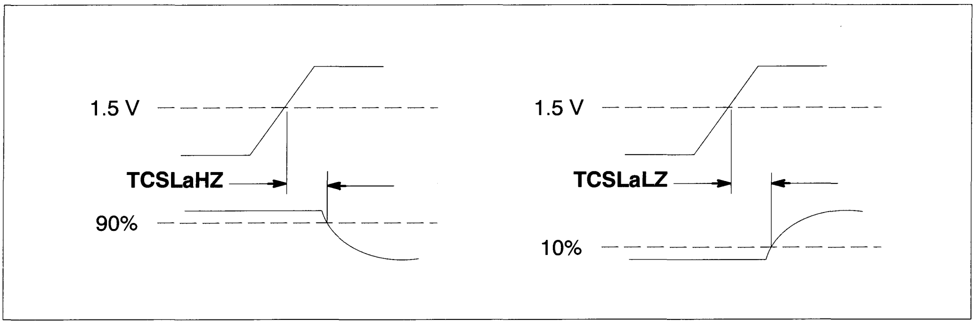


Figure 6.4 IMS C012 tristate timing relative to **notCS**

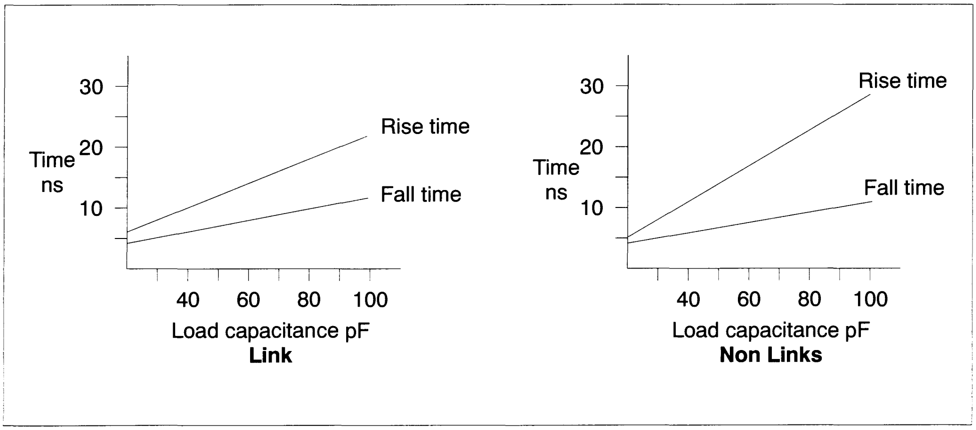


Figure 6.5 Typical rise/fall times

6.4 Power rating

Internal power dissipation (P_{INT}) of transputer and peripheral chips depends on **VDD**, as shown in figure 6.6. P_{INT} is substantially independent of temperature.

Total power dissipation (P_D) of the chip is

$$P_D = P_{INT} + P_{IO}$$

where P_{IO} is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature T_J of the chip is

$$T_J = T_A + \theta_{JA} * P_D$$

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W. θ_{JA} for each package is given in Appendix A, Packaging Specifications.

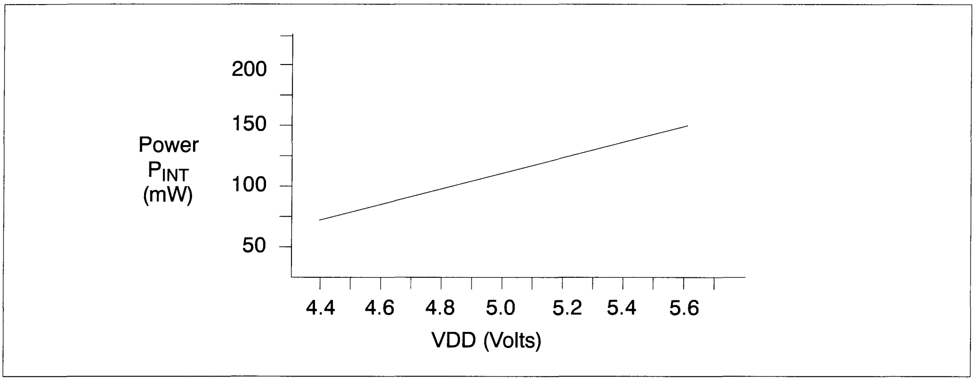


Figure 6.6 IMS C012 internal power dissipation vs VDD

7 Package pinouts

7.1 24 pin dual-in-line package

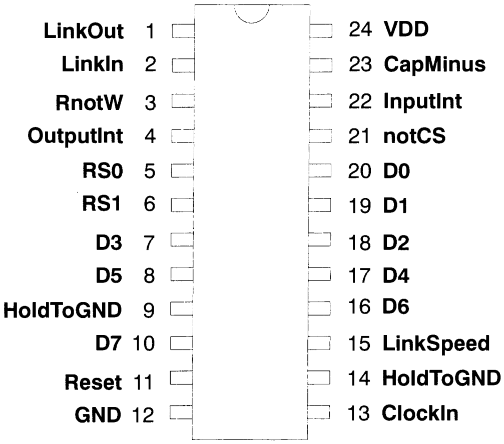


Figure 7.1 IMS C012 24 pin DIL package pinout

8 Ordering

This section indicates the designation of package selections for the IMS C012. Speed of **ClockIn** is 5 MHz for all parts.

For availability contact your local SGS–THOMSON sales office or authorized distributor.

INMOS designation	Package
IMS C012-P20S	24 pin plastic dual-in-line

Table 8.1 IMS C012 ordering details

Packaging specifications

1 24 pin plastic dual-in-line (DIL) package dimensions

DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	–	–	0.195	–	–	4.953
A1	0.015	–	–	0.381	–	–
A2	–	0.130	–	–	3.302	–
B	0.016	0.018	0.020	0.406	0.457	0.508
B1	0.045	0.055	0.065	1.143	1.397	1.651
C	0.008	0.010	0.012	0.203	0.254	0.305
D	1.240	1.250	1.270	31.496	31.750	32.258
E	0.297	–	0.325	7.544	–	8.255
E1	0.240	0.260	0.290	6.096	6.604	7.366
e	–	0.100BSC	–	–	2.540BSC	–
eA	–	0.300TYP	–	–	7.620TYP	–
e3	–	1.100BSC	–	–	27.940BSC	–
L	0.120	0.130	0.150	3.048	3.302	3.810
M	0 DEG	–	15 DEG	0 RAD	–	0.26 RAD
S	0.070	0.075	0.085	1.778	1.905	2.159

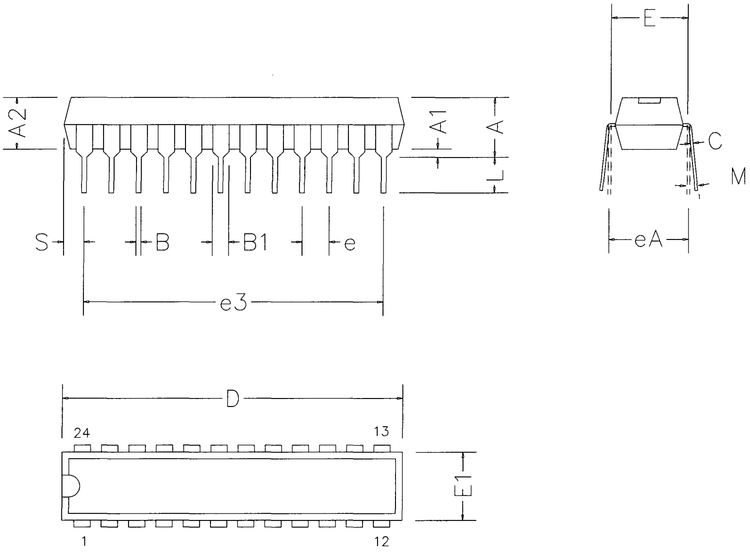


Figure 1.1 24 pin plastic dual-in-line package dimensions

2 28 pin plastic dual-in-line (DIL) package dimensions

DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	–	–	0.195	–	–	4.953
A1	0.015	–	–	0.381	–	–
A2	0.120	–	0.180	3.048	–	4.572
B	0.014	0.018	0.022	0.356	0.457	0.559
B1	0.045	0.055	0.065	1.143	1.397	1.651
C	0.008	0.010	0.012	0.203	0.254	0.305
D	1.440	1.450	1.470	36.576	36.830	37.338
E	0.597	–	0.625	15.164	–	15.875
E1	0.530	0.550	0.560	13.462	13.970	14.224
e	–	0.100BSC	–	–	2.540BSC	–
eA	–	0.600TYP	–	–	15.240TYP	–
e3	–	1.300BSC	–	–	33.020BSC	–
L	0.125	0.130	0.135	3.175	3.302	3.429
M	0 DEG	–	15 DEG	0 RAD	–	0.26 RAD
S	0.070	0.075	0.080	1.778	1.905	2.032

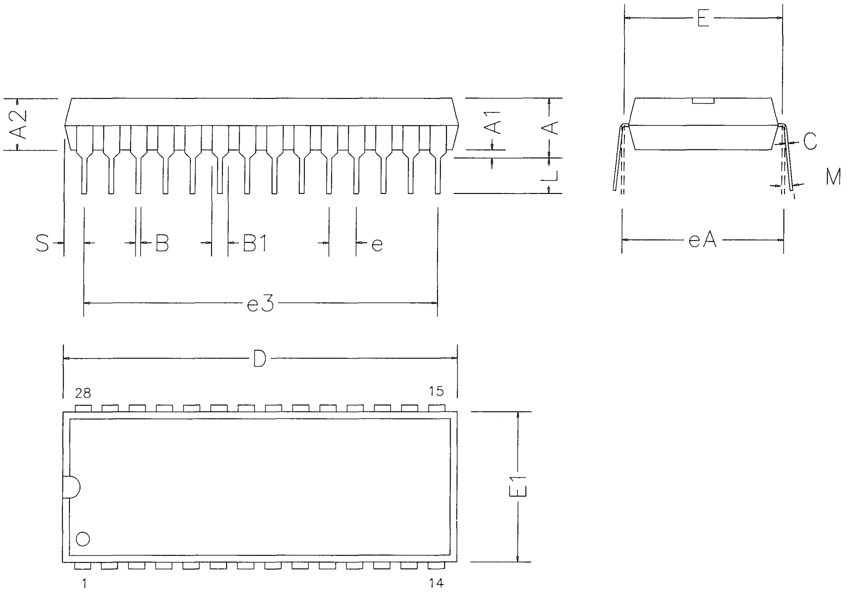


Figure 2.1 28 pin plastic dual-in-line package dimensions

3 28 pin ceramic dual-in-line (DIL) package dimensions

	Inches		Millimetres	
DIM	Min	Max	Min	Max
A	0.085	0.110	2.159	2.794
A1	0.035	0.065	0.889	1.651
B	0.016	0.022	0.406	0.559
B1	0.044	0.056	1.118	1.422
D	1.384	1.416	35.154	35.966
E	0.604	0.638	15.342	16.205
E1	0.595	0.625	15.113	15.875
e1	0.090	0.110	2.286	2.794
L	0.125	0.165	3.175	4.191

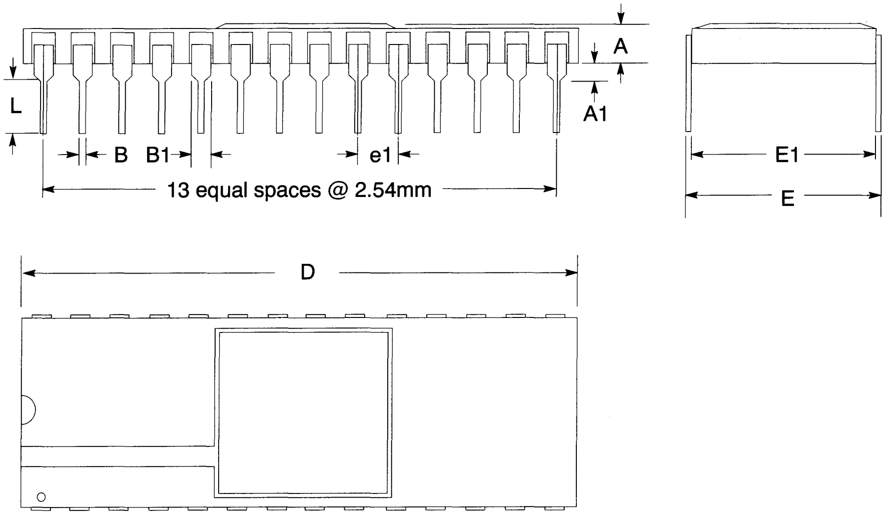


Figure 3.1 28 pin ceramic dual-in-line package dimensions

4 28 pin plastic small outline J-leaded (SOJ) package dimensions

	Inches		Millimetres	
DIM	Min	Max	Min	Max
A	0.120	0.140	3.048	3.556
B	0.014	0.019	0.356	0.483
C	0.009	0.013	0.229	0.331
D	0.701	0.711	17.805	18.060
E	0.335	0.347	8.509	8.814
E1	0.292	0.299	7.417	7.595
e	0.050TYP		1.270TYP	
eA	0.262	0.272	6.655	6.910
A1	0.028	0.036	0.711	0.914

Coplanarity 0.1 mm Max.

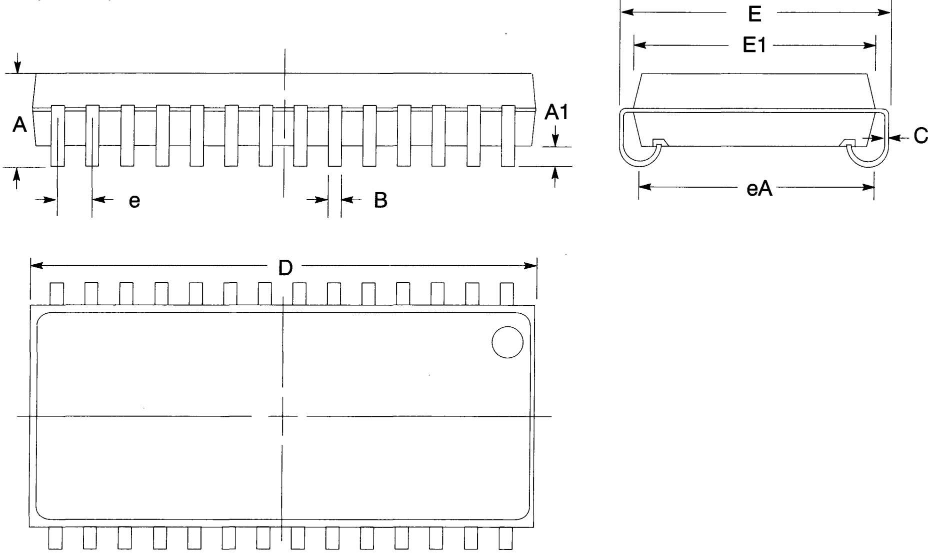


Figure 4.1 28 pin plastic small outline J-leaded package dimensions

5 28 pin leadless chip carrier (LCC) package dimensions

	Inches		Millimetres	
DIM	Min	Max	Min	Max
A	0.064	0.078	1.625	1.981
B1	0.022	0.028	0.559	0.711
D	0.540	0.560	13.716	14.224
E	0.340	0.360	8.636	9.144
e1	0.048	0.052	1.219	1.321

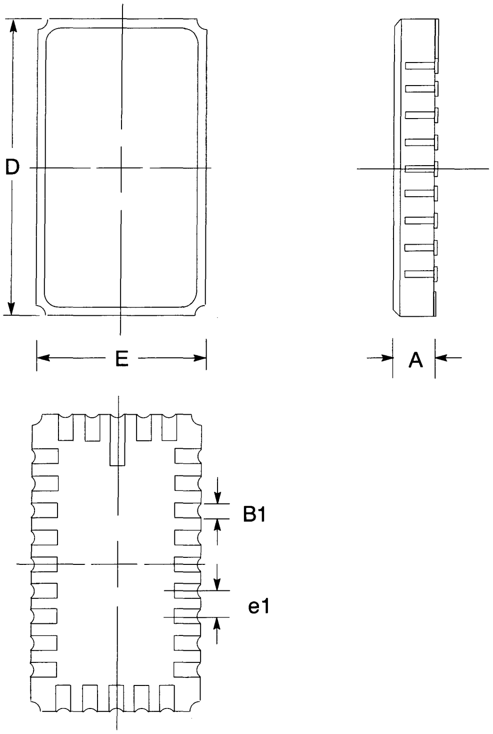


Figure 5.1 28 pin leadless chip carrier package dimensions

6 68 pin grid array (PGA) package dimensions

DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.135	0.147	0.160	3.429	3.734	4.064
A1	–	0.050REF	–	–	1.270REF	–
A2	0.085	0.097	0.110	2.159	2.464	2.794
B	–	0.018	–	–	0.457	–
B1	–	0.050REF	–	–	1.270REF	–
D	1.050	1.060	1.070	35.179	35.560	36.068
D1	–	0.900REF	–	–	22.86REF	–
e	–	0.100BSC	–	–	2.540BSC	–
E	1.050	1.060	1.070	35.179	35.560	36.068
E1	–	0.900REF	–	–	22.86REF	–
L	–	0.130	–	–	3.302	–

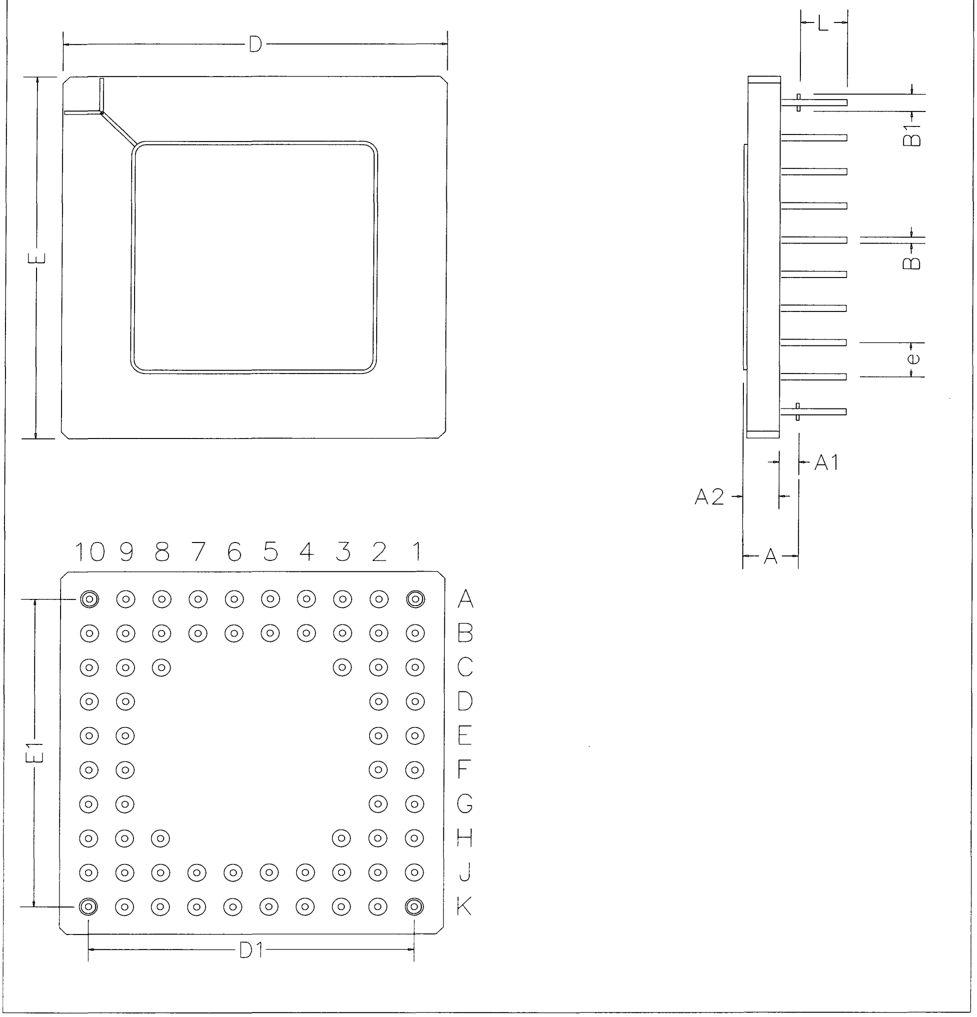
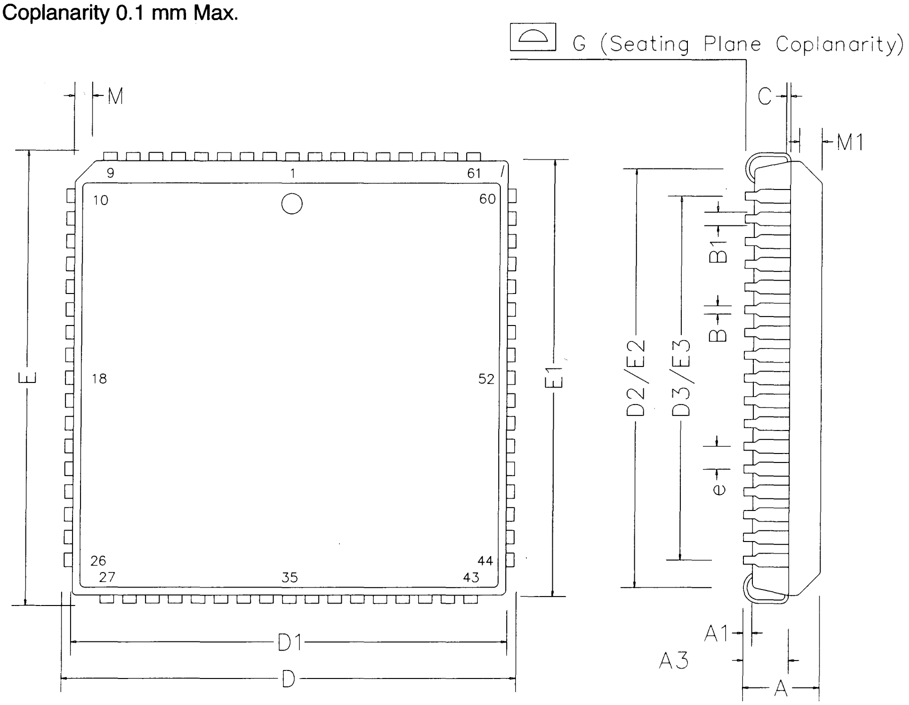


Figure 6.1 68 pin grid array package dimensions

7 68 pin plastic leadless chip carrier (PLCC) J-bend package dimensions

DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.165	0.170	0.200	4.191	4.318	5.080
A1	0.020	–	–	0.508	–	–
A3	0.090	–	0.130	2.290	–	3.302
B	0.013	–	0.023	0.330	–	0.584
B1	0.025	–	0.035	0.635	–	0.889
C	0.0075	0.008	0.0085	0.191	0.203	0.216
D	0.985	0.990	0.995	25.019	25.146	25.273
D1	0.950	0.955	0.960	24.130	24.257	24.384
D2	0.890	0.910	0.930	22.606	23.114	23.622
D3	–	0.800REF	–	–	20.320REF	–
E	0.985	0.990	0.995	25.019	25.146	25.273
E1	0.950	0.955	0.960	24.130	24.257	24.384
E2	0.890	0.910	0.930	22.606	23.114	23.622
E3	–	0.800REF	–	–	20.320REF	–
e	–	0.050BSC	–	–	1.270BSC	–
G	–	–	0.004	–	–	0.102
L	–	N/A	–	–	N/A	–
L1	–	N/A	–	–	N/A	–
M	0.042	–	0.048	1.067	–	1.219
M1	0.042	–	0.056	1.067	–	1.422

Coplanarity 0.1 mm Max.



Notes;

1. Maximum lead displacement from notional centre line = ±0.007".

Figure 7.1 68 pin PLCC J-bend package dimensions

8 84 pin grid array (PGA) package dimensions

DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.135	0.147	0.160	3.429	3.734	4.064
A1	–	0.050REF	–	–	1.270REF	–
A2	0.085	0.097	0.110	2.159	2.464	2.794
B	–	0.018	–	–	0.457	–
B1	–	0.050REF	–	–	1.270REF	–
D	1.050	1.060	1.070	35.179	35.560	36.068
D1	–	0.900REF	–	–	22.86REF	–
e	–	0.100BSC	–	–	2.540BSC	–
E	1.050	1.060	1.070	35.179	35.560	36.068
E1	–	0.900REF	–	–	22.86REF	–
L	–	0.130	–	–	3.302	–

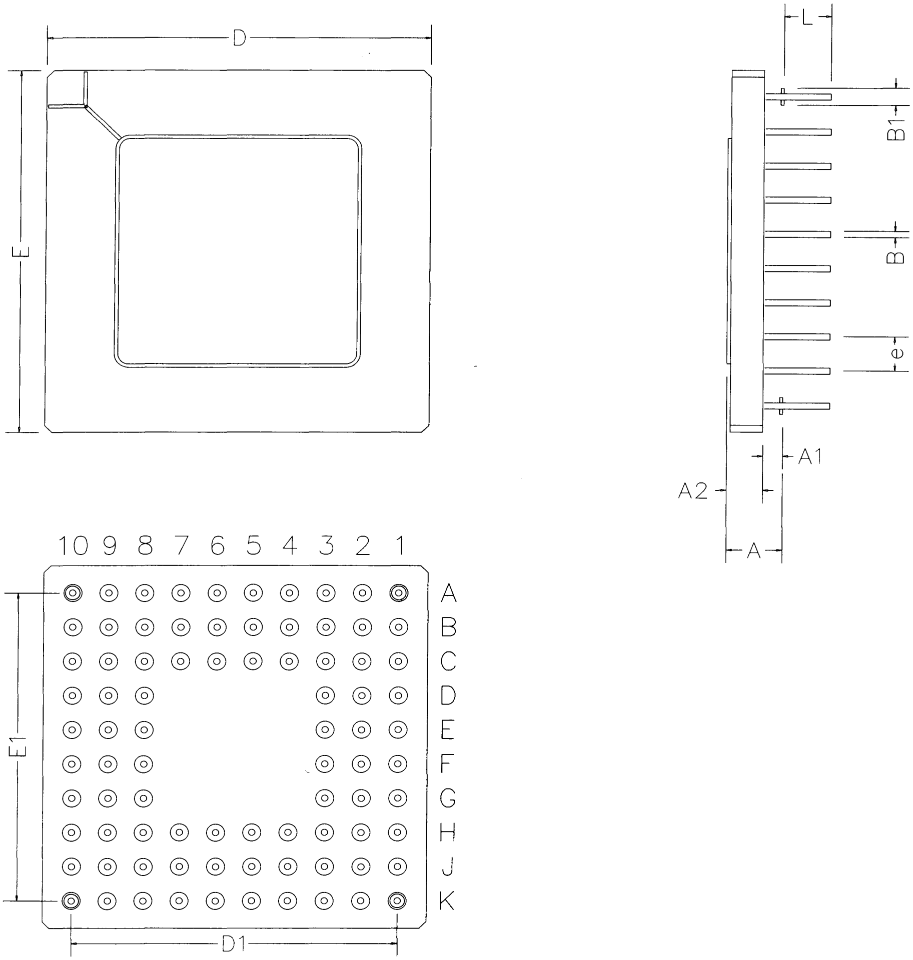
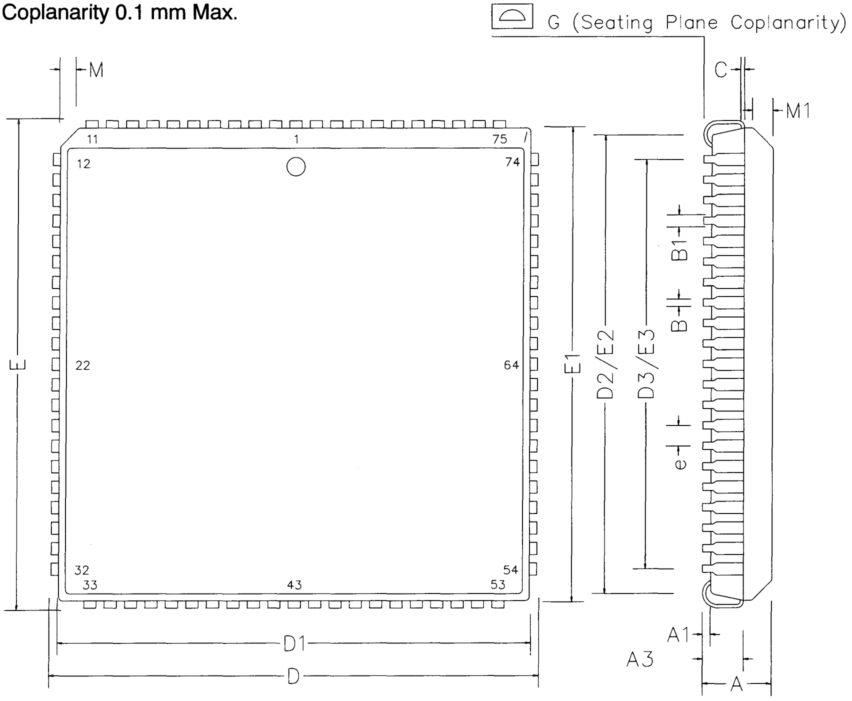


Figure 8.1 84 pin grid array package dimensions

9 84 pin plastic leadless chip carrier (PLCC) J-bend package dimensions

DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.165	0.170	0.200	4.191	4.318	5.080
A1	0.020	–	–	0.508	–	–
A3	0.090	–	0.130	2.290	–	3.302
B	0.013	–	0.023	0.330	–	0.584
B1	0.025	–	0.035	0.635	–	0.889
C	0.0075	0.008	0.0085	0.191	0.203	0.216
D	1.185	1.190	1.195	30.099	30.266	30.353
D1	1.149	1.154	1.159	29.185	29.312	29.439
D2	1.090	1.120	1.130	27.686	28.488	28.702
D3	–	1.000REF	–	–	25.400REF	–
E	1.185	1.190	1.195	30.099	30.266	30.353
E1	1.150	1.155	1.159	29.285	29.312	29.439
E2	1.090	1.120	1.130	27.686	28.488	28.702
E3	–	1.000REF	–	–	25.400REF	–
e	–	0.050BSC	–	–	1.270BSC	–
G	–	–	0.004	–	–	0.102
L	–	N/A	–	–	N/A	–
L1	–	N/A	–	–	N/A	–
M	0.042	–	0.048	1.067	–	1.219
M1	0.042	–	0.056	1.067	–	1.422

Coplanarity 0.1 mm Max.



Notes;

1. Maximum lead displacement from notional centre line = ±0.007”.

Figure 9.1 84 pin PLCC J-bend package dimensions

10 100 pin ceramic quad flat pack (CQFP) package dimensions

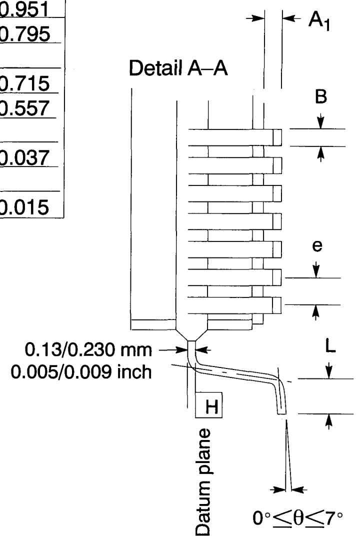
10.1 100 pin cavity-down ceramic quad flat pack

	CONTROL DIMENSIONS mm			ALTERNATIVE DIMENSIONS INCH		
DIM	Min	Nom	Max	Min	Nom	Max
A			3.400			0.134
A_1	0.250			0.010		
A_2			3.073			0.121
D	23.65	23.90	24.15	0.931	0.941	0.951
D_1	19.80	20.00	20.20	0.780	0.787	0.795
D_3		18.85 REF			0.742 REF	
E	17.65	17.90	18.15	0.695	0.705	0.715
E_1	13.84	14.00	14.15	0.545	0.551	0.557
E_3		12.35 REF			0.486 REF	
L	0.650	0.800	0.950	0.026	0.031	0.037
e		0.650			0.026 BSC	
B	0.22	BSC	0.38	0.009		0.015

Notes

1 Dimension D1 and E1 measured at the ceramic side. Base to window frame misalignment is 0.40mm max. An additional glass meniscus of 0.2mm max is allowed at the leadframe interface.

2 Coplanarity 0.1 mm Max.



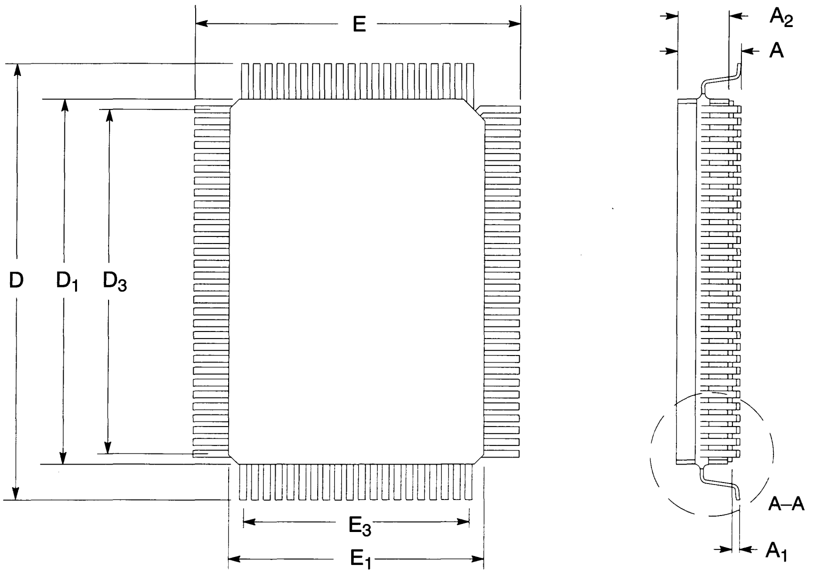


Figure 10.1 100 pin cavity-down ceramic quad flat pack dimensions

10.2 100 pin cavity-up ceramic quad flat pack

	CONTROL DIMENSIONS mm			ALTERNATIVE DIMENSIONS INCH		
DIM	**Min**	**Nom**	**Max**	**Min**	**Nom**	**Max**
A			3.400			0.134
A_1	0.250			0.010		
A_2			3.073			0.121
D	23.65	23.90	24.15	0.931	0.941	0.951
D_1	19.80	20.00	20.20	0.780	0.787	0.795
D_3		18.85 REF			0.742 REF	
E	17.65	17.90	18.15	0.695	0.705	0.715
E_1	13.84	14.00	14.15	0.545	0.551	0.557
E_3		12.35 REF			0.486 REF	
L	0.650	0.800	0.950	0.026	0.031	0.037
e		0.650 BSC			0.026 BSC	
B	0.22		0.38	0.009		0.015

Notes

1 Dimension D1 and E1 measured at the ceramic side. Base to window frame misalignment is 0.40mm max. An additional glass meniscus of 0.2mm max is allowed at the leadframe interface.

2 Coplanarity 0.1 mm Max

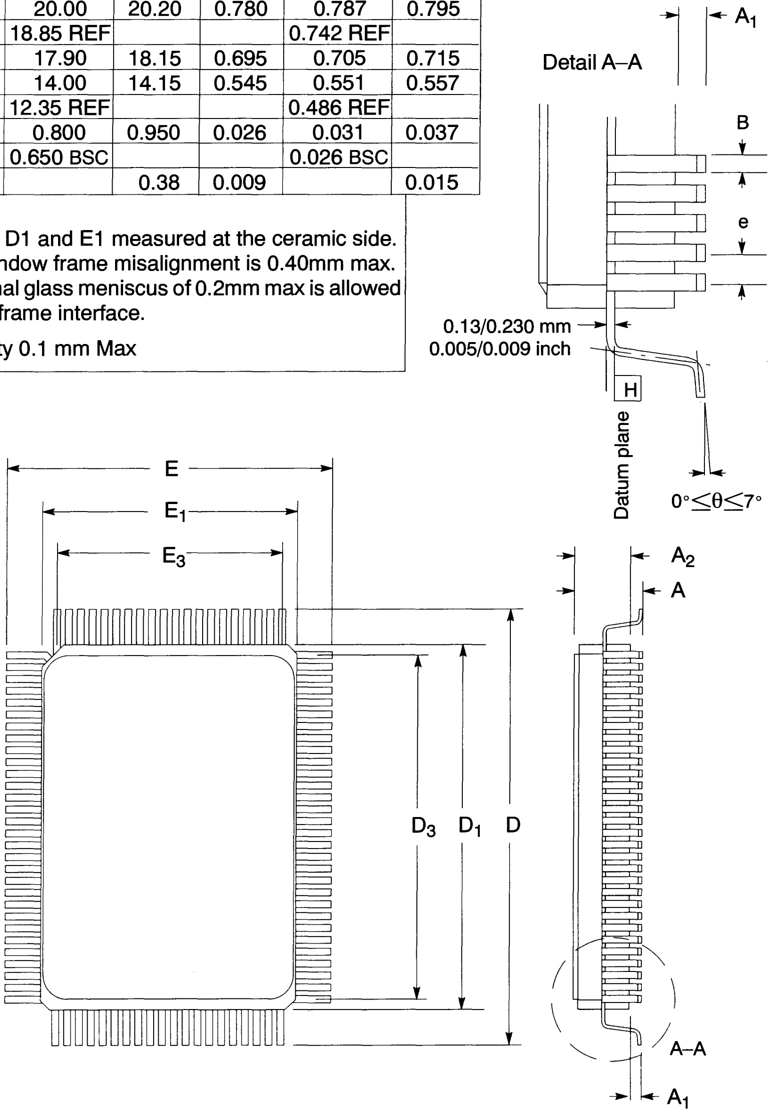


Figure 10.2 100 pin cavity-up ceramic quad flat pack dimensions

11 100 pin plastic quad flat pack (PQFP) package dimensions

DIM	Millimetres		Inches		Notes
	Min	Max	Min	Max	
A		3.4		0.134	
A1	0.1	0.36	0.004	0.014	
A2	2.55	3.05	0.1	0.120	
D	22.95	24.15	0.904	0.951	1
D1	19.90	20.10	0.783	0.791	2
D3	18.85REF		0.742REF		
Z_D	0.58REF		0.023REF		
E	16.95	18.15	0.667	0.715	1
E1	13.90	14.10	0.547	0.555	2
E3	12.35REF		0.486REF		
Z_E	0.83REF		0.033REF		
L	0.65	0.95	0.026	0.037	
e	0.65 BSC		0.026BSC		
W	0.22	0.38	0.009	0.015	
θ	0°	7°	0°	7°	

Notes

1 To be determined at seating plane C.

2 To be determined at datum plane H.

3 Coplanarity 0.1 mm Max.

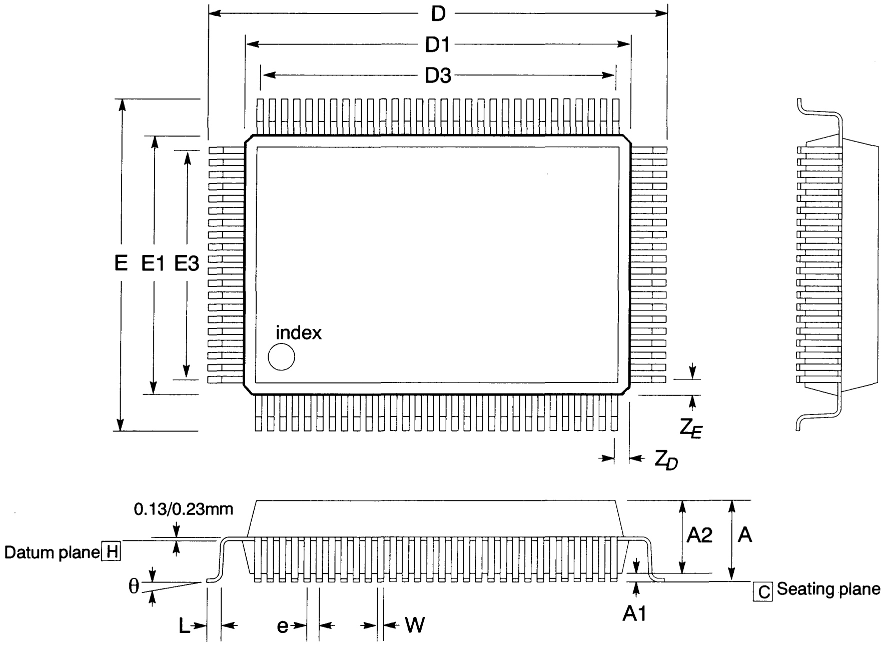


Figure 11.1 100 pin plastic quad flat pack dimensions

12 100 pin grid array (PGA) package dimensions

DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.135	0.147	0.160	3.429	3.734	4.064
A1	–	0.050REF	–	–	1.270REF	–
A2	0.085	0.097	0.110	2.159	2.464	2.794
B	–	0.018	–	–	0.457	–
B1	–	0.050REF	–	–	1.270REF	–
D	1.050	1.060	1.070	35.179	35.560	36.068
D1	–	0.900REF	–	–	22.86REF	–
e	–	0.100BSC	–	–	2.540BSC	–
E	1.050	1.060	1.070	35.179	35.560	36.068
E1	–	0.900REF	–	–	22.86REF	–
L	–	0.130	–	–	3.302	–

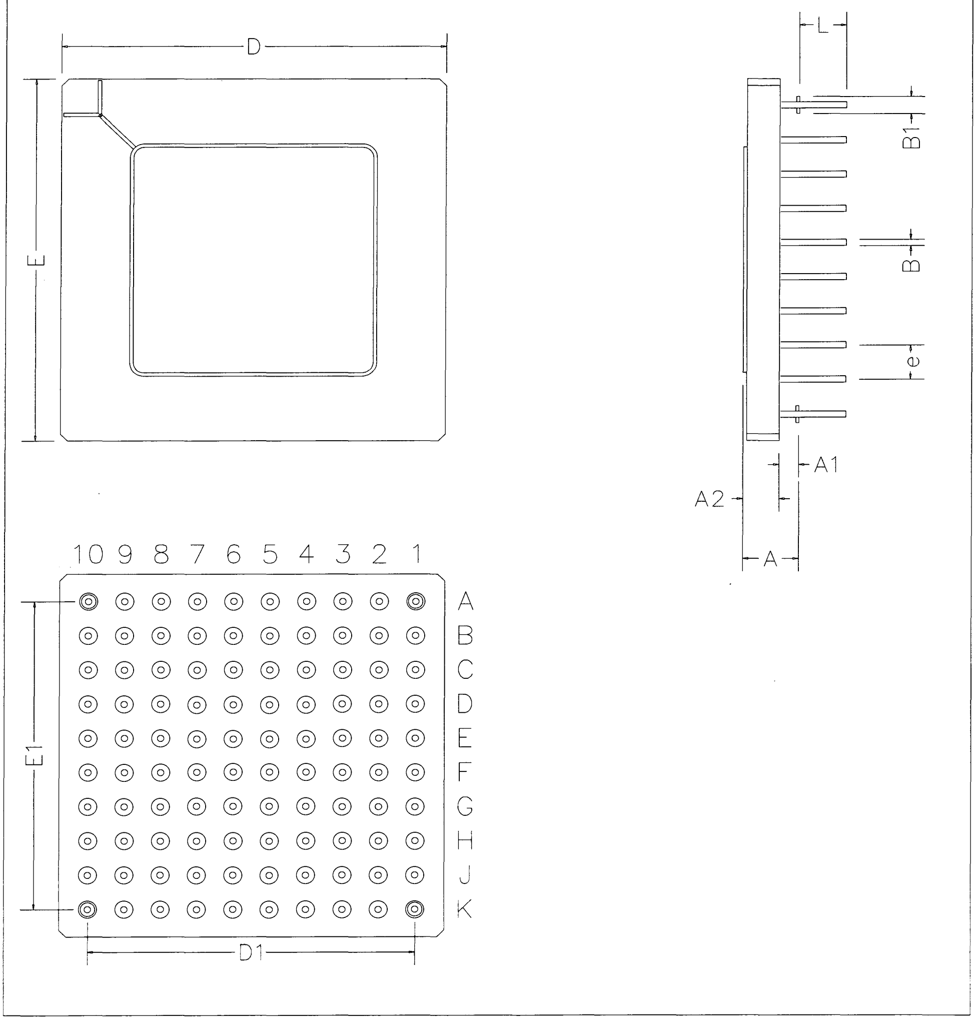


Figure 13.1 100 pin grid array package dimensions

13 Package thermal characteristics

Package	θ_{JA} **(°C/W)**
24 pin plastic dual-in-line	115
28 pin plastic dual-in-line	110
28 pin ceramic dual-in-line	60
28 pin plastic small outline J-leaded (SOJ)	140
28 pin leadless chip carrier (LCC)	120
68 pin grid array	35
68 pin PLCC J-bend	35
84 pin grid array	35
84 pin PLCC J-bend	35
100 pin ceramic quad flat pack	49
100 pin plastic quad flat pack	42
100 pin grid array	35

Table 13.1 Package thermal characteristics
(at 400 linear ft/min transverse air flow)

Full details are contained in the following SGS-THOMSON catalogues, available from SGS-THOMSON sales offices and authorized distributors worldwide.

1 *Thermal Management in Surface Mount Technology,* order code: BRTHERMAN/0788

2 *Reliability in Surface Mount Technology,* order code: BRRELSMT/1088

Obsolete devices

1 Introduction

This section contains a summary of devices that have been replaced by newer family members.

As part of INMOS's continual program and product improvement enhancements to the transputer family have been made whilst maintaining compatibility within the family.

These changes are primarily moving from 1.5μ to 1.2μ CMOS enabling faster parts to be produced and the addition of new functions such as breakpoint instructions and event waiting / refresh pending pins.

The IMS M212 device has been withdrawn with no replacement since it was designed to support the disk interface ST506 whilst the market has now moved primarily to the SCSI standard.

Obsolete devices and their replacements are as follows:

- T800 – replaced by T805
- T414 – replaced by T425 or T400
- T212 – replaced by T222
- M212 – no replacement

IMS T800 transputer

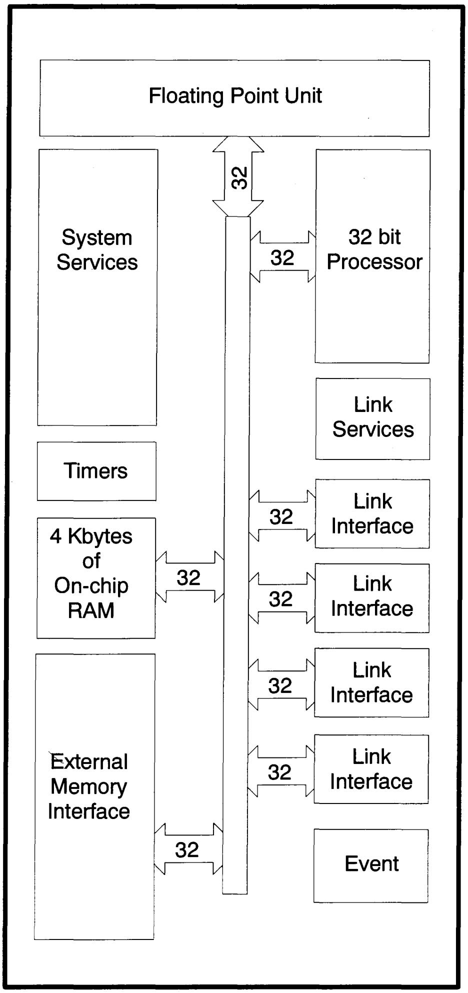
Replaced by IMS T805

inmos®

Engineering Data

FEATURES

32 bit architecture
33 ns internal cycle time
30 MIPS (peak) instruction rate
4.3 Mflops (peak) instruction rate
64 bit on-chip floating point unit which conforms to IEEE 754
4 Kbytes on-chip static RAM
120 Mbytes/sec sustained data rate to internal memory
4 Gbytes directly addressable external memory
40 Mbytes/sec sustained data rate to external memory
630 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
Bi-directional data rate of 2.4 Mbytes/sec per link
High performance graphics support with block move instructions
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ±5% power supply
Packaging 84 pin PGA



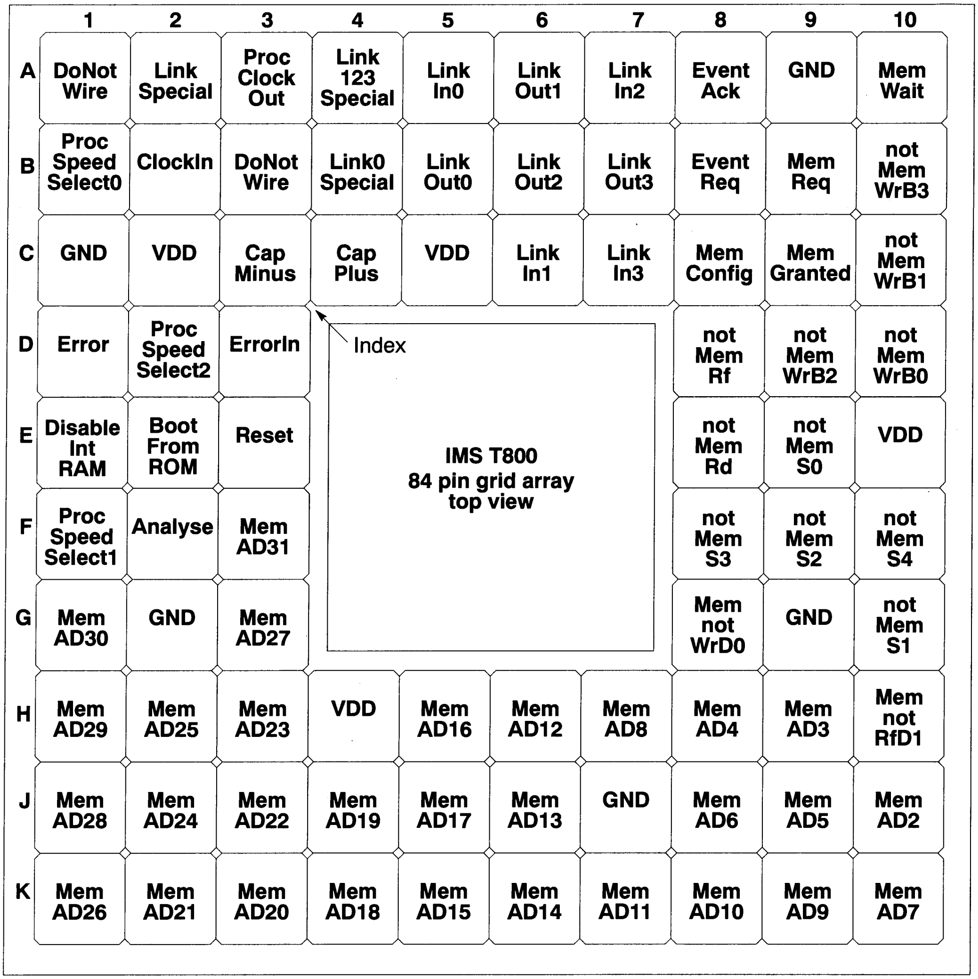
APPLICATIONS

Scientific and mathematical applications
High speed multi processor systems
High performance graphics processing
Supercomputers
Workstations and workstation clusters
Digital signal processing
Accelerator processors
Distributed databases
System simulation
Telecommunications
Robotics
Fault tolerant systems
Image processing
Pattern recognition
Artificial intelligence

SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

1.1 Package specifications

1.1.1 84 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	DoNot Wire	Link Special	Proc Clock Out	Link 123 Special	Link In0	Link Out1	Link In2	Event Ack	GND	Mem Wait
B	Proc Speed Select0	ClockIn	DoNot Wire	Link0 Special	Link Out0	Link Out2	Link Out3	Event Req	Mem Req	not Mem WrB3
C	GND	VDD	Cap Minus	Cap Plus	VDD	Link In1	Link In3	Mem Config	Mem Granted	not Mem WrB1
D	Error	Proc Speed Select2	ErrorIn					not Mem Rf	not Mem WrB2	not Mem WrB0
E	Disable Int RAM	Boot From ROM	Reset					not Mem Rd	not Mem S0	VDD
F	Proc Speed Select1	Analyse	Mem AD31					not Mem S3	not Mem S2	not Mem S4
G	Mem AD30	GND	Mem AD27					Mem not WrD0	GND	not Mem S1
H	Mem AD29	Mem AD25	Mem AD23	VDD	Mem AD16	Mem AD12	Mem AD8	Mem AD4	Mem AD3	Mem not RfD1
J	Mem AD28	Mem AD24	Mem AD22	Mem AD19	Mem AD17	Mem AD13	GND	Mem AD6	Mem AD5	Mem AD2
K	Mem AD26	Mem AD21	Mem AD20	Mem AD18	Mem AD15	Mem AD14	Mem AD11	Mem AD10	Mem AD9	Mem AD7

Figure 1 .1 IMS T800 84 pin grid array

inmos®

IMS T414 transputer

Replaced by IMS T425

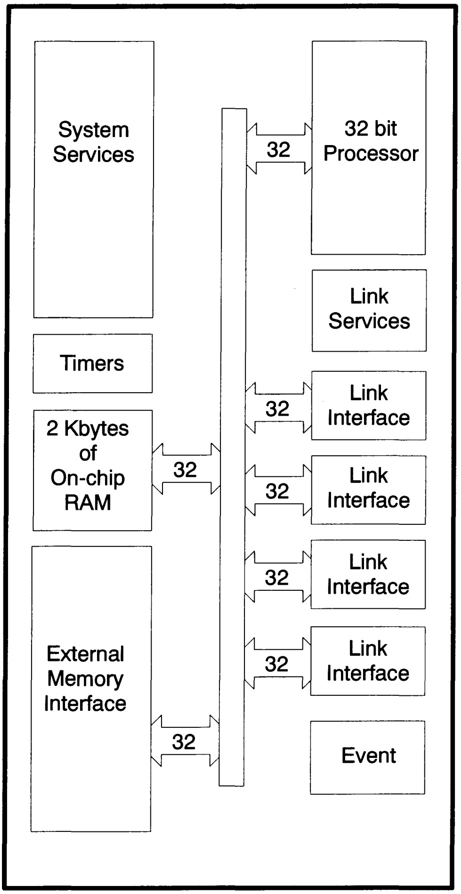
Engineering Data

FEATURES

32 bit architecture
50 ns internal cycle time
20 MIPS peak instruction rate
Pin compatible with IMS T805-20, IMS T800-20, and IMS T425-20
2 Kbytes on-chip static RAM
80 Mbytes/sec sustained data rate to internal memory
4 Gbytes directly addressable external memory
26 Mbytes/sec sustained data rate to external memory
950 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
Bi-directional data rate 0f 1.6 Mbytes/sec per link
Internal timers of 1 μs and 64 μs
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ± 5% power supply
Packaging 84 pin PGA / 84 pin PLCC

APPLICATIONS

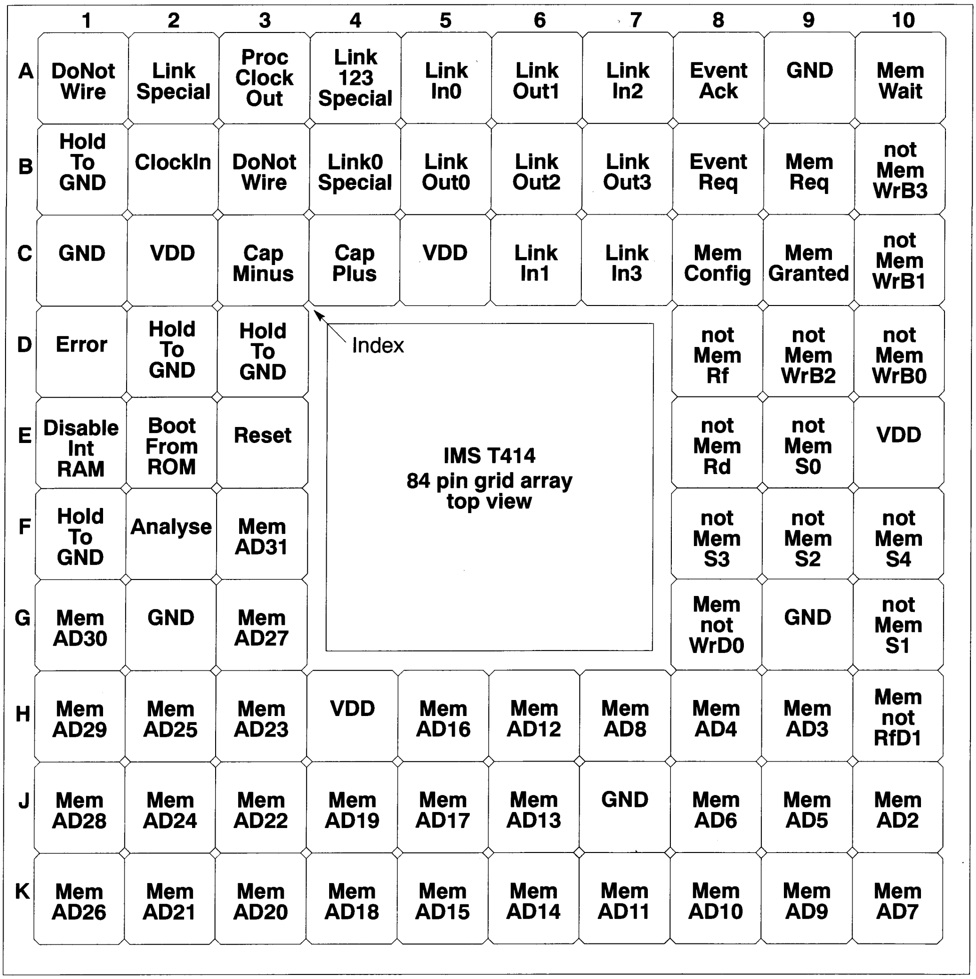
High speed multi processor systems
Real time processing
Microprocessor applications
Workstations and workstation clusters
Image processing
Graphics processing
Accelerator processors
Distributed databases
Supercomputers
System simulation
Digital signal processing
Telecommunications
Robotics
Fault tolerant systems
Medical instrumentation
Pattern recognition
Artificial intelligence



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

2.1 Package specifications

2.1.1 84 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	DoNot Wire	Link Special	Proc Clock Out	Link 123 Special	Link In0	Link Out1	Link In2	Event Ack	GND	Mem Wait
B	Hold To GND	ClockIn	DoNot Wire	Link0 Special	Link Out0	Link Out2	Link Out3	Event Req	Mem Req	not Mem WrB3
C	GND	VDD	Cap Minus	Cap Plus	VDD	Link In1	Link In3	Mem Config	Mem Granted	not Mem WrB1
D	Error	Hold To GND	Hold To GND					not Mem Rf	not Mem WrB2	not Mem WrB0
E	Disable Int RAM	Boot From ROM	Reset					not Mem Rd	not Mem S0	VDD
F	Hold To GND	Analyse	Mem AD31					not Mem S3	not Mem S2	not Mem S4
G	Mem AD30	GND	Mem AD27					Mem not WrD0	GND	not Mem S1
H	Mem AD29	Mem AD25	Mem AD23	VDD	Mem AD16	Mem AD12	Mem AD8	Mem AD4	Mem AD3	Mem not RfD1
J	Mem AD28	Mem AD24	Mem AD22	Mem AD19	Mem AD17	Mem AD13	GND	Mem AD6	Mem AD5	Mem AD2
K	Mem AD26	Mem AD21	Mem AD20	Mem AD18	Mem AD15	Mem AD14	Mem AD11	Mem AD10	Mem AD9	Mem AD7

Figure 2 .1 IMS T414 84 pin grid array

2.1.2 84 pin PLCC J-bend package

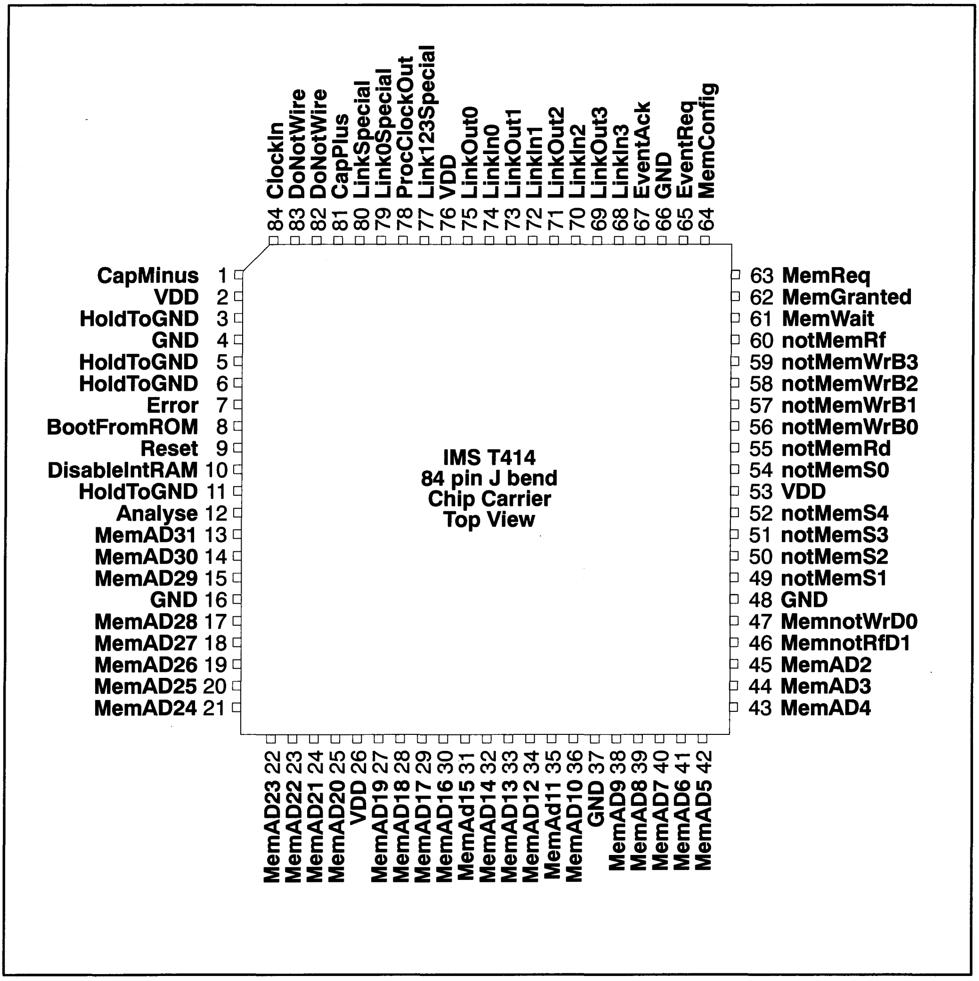


Figure 2 .2 IMS T414 84 pin PLCC J-bend package pinout

IMS T212 transputer

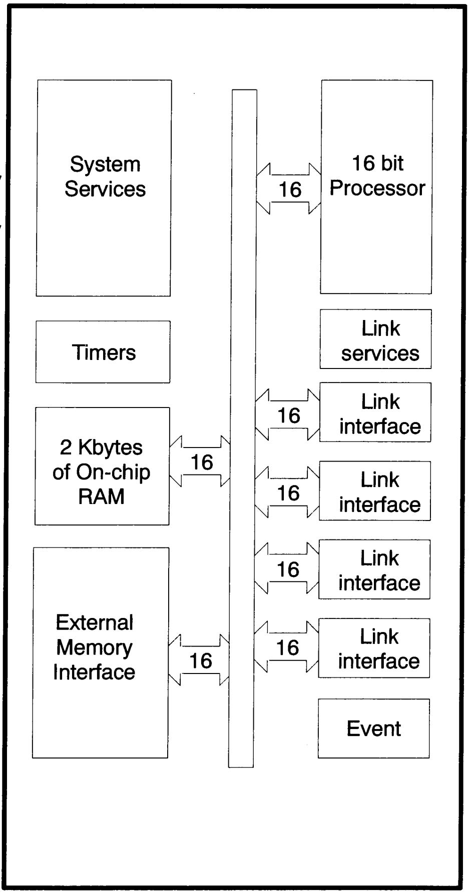
Replaced by IMS T225

inmos®

Engineering Data

FEATURES

16 bit architecture
50 ns internal cycle time
20 MIPS (peak) instruction rate
Pin compatible with IMS T222
2 Kbytes on-chip static RAM
80 Mbytes/sec sustained data rate to internal memory
64 Kbytes directly addressable external memory
20 Mbytes/sec sustained data rate to external memory
950 ns response to interrupts
Four INMOS serial links 5/10/20 Mbits/sec
Bi-directional data rate of 1.6 Mbytes/sec per link
Internal timers of 1 μs and 64 μs
Boot from ROM or communication links
Single 5 MHz clock input
Single +5V ±5% power supply, less than 1 Watt
Packaging 68 pin PGA



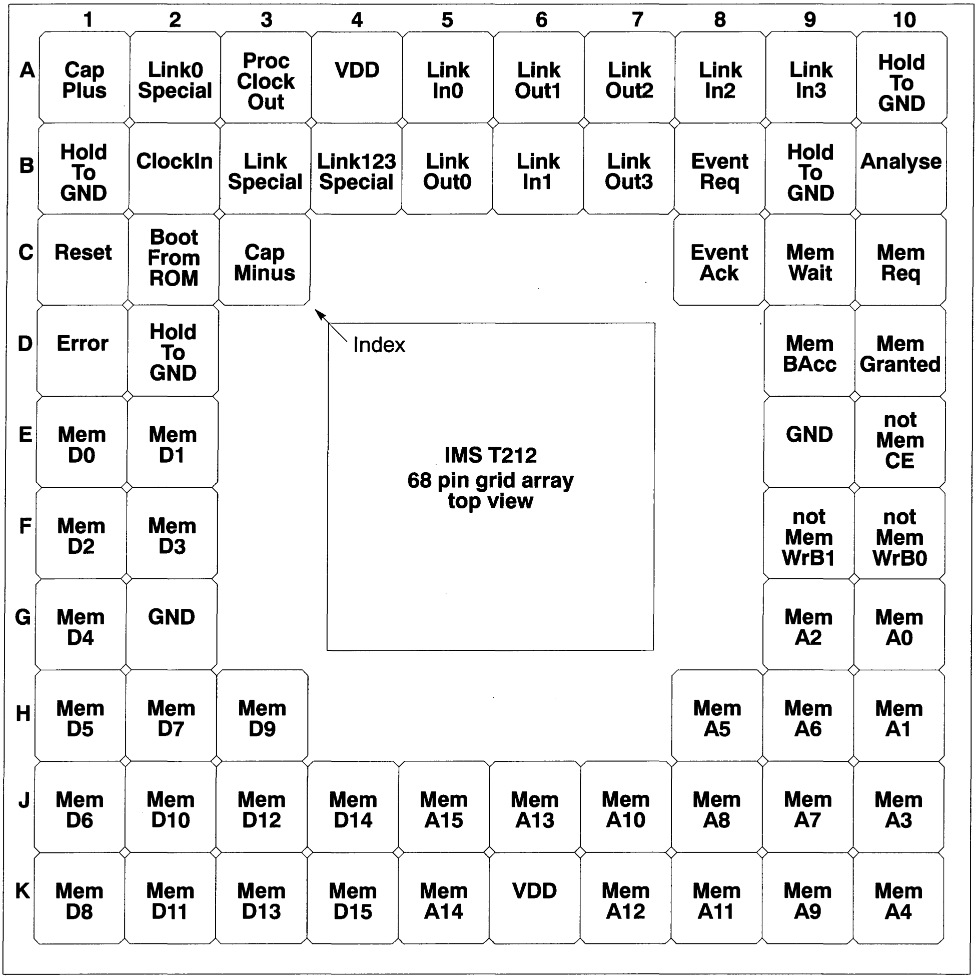
APPLICATIONS

Real time processing
Microprocessor applications
High speed multi processor systems
Industrial control
Robotics
System simulation
Digital signal processing
Telecommunications
Fault tolerant systems
Medical instrumentation
Pattern recognition
Image processing
Graphics processing
Artificial intelligence
Supercomputers

SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

3.1 Package specifications

3.1.1 68 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Cap Plus	Link0 Special	Proc Clock Out	VDD	Link In0	Link Out1	Link Out2	Link In2	Link In3	Hold To GND
B	Hold To GND	ClockIn	Link Special	Link123 Special	Link Out0	Link In1	Link Out3	Event Req	Hold To GND	Analyse
C	Reset	Boot From ROM	Cap Minus					Event Ack	Mem Wait	Mem Req
D	Error	Hold To GND							Mem BAcc	Mem Granted
E	Mem D0	Mem D1							GND	not Mem CE
F	Mem D2	Mem D3							not Mem WrB1	not Mem WrB0
G	Mem D4	GND							Mem A2	Mem A0
H	Mem D5	Mem D7	Mem D9					Mem A5	Mem A6	Mem A1
J	Mem D6	Mem D10	Mem D12	Mem D14	Mem A15	Mem A13	Mem A10	Mem A8	Mem A7	Mem A3
K	Mem D8	Mem D11	Mem D13	Mem D15	Mem A14	VDD	Mem A12	Mem A11	Mem A9	Mem A4

Figure 3 .1 IMS T212 68 pin grid array package pinout

IMS M212 disk processor

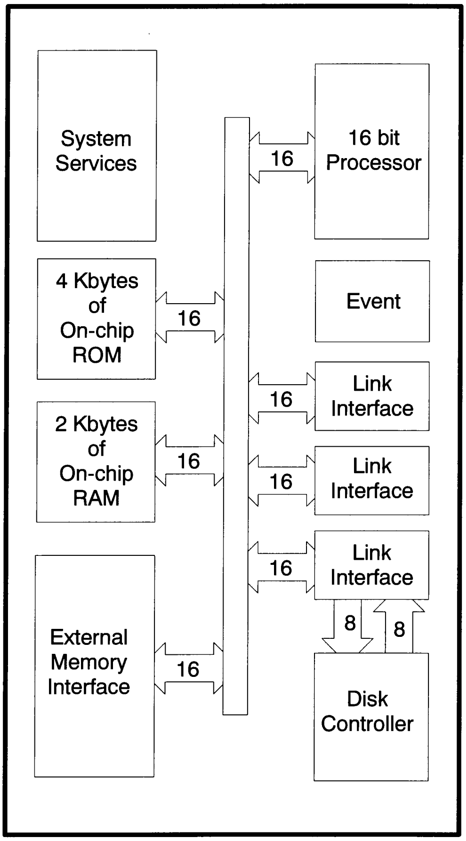
No replacement

inmos®

Product Preview

FEATURES

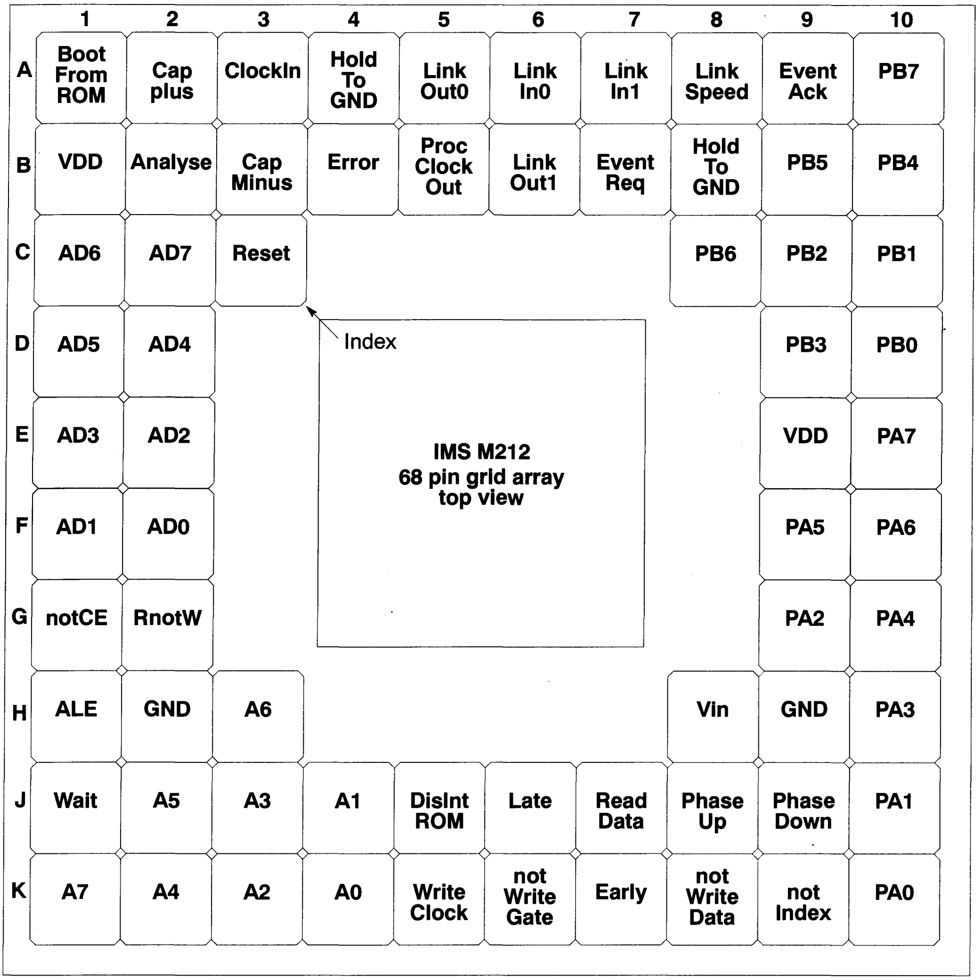
ST506/ST412, SA400/450 compatible interface
Full disk interface logic on chip
Minimum of external components required
On-chip 16 bit processor
2 Kbytes on-chip RAM
4 Kbytes on-chip ROM disk control software
External memory interface
Hardware CRC/ECC generator
Two bi-directional 8 bit data ports
Two INMOS serial links 10/20 Mbits/sec
External event interrupt
Variable wait states for slow memory
Internal timers
Support for run-time error diagnostics
Bootstraps from ROM, link or disk
Single 5 MHz processor clock input
Power dissipation less than 1 Watt
Packaging 68 pin PGA



SGS-THOMSON MICROELECTRONICS
INMOS is a member of the SGS–THOMSON Microelectronics group

4.1 Package specifications

4.1.1 68 pin grid array package



	1	2	3	4	5	6	7	8	9	10
A	Boot From ROM	Cap plus	ClockIn	Hold To GND	Link Out0	Link In0	Link In1	Link Speed	Event Ack	PB7
B	VDD	Analyse	Cap Minus	Error	Proc Clock Out	Link Out1	Event Req	Hold To GND	PB5	PB4
C	AD6	AD7	Reset					PB6	PB2	PB1
D	AD5	AD4							PB3	PB0
E	AD3	AD2							VDD	PA7
F	AD1	AD0							PA5	PA6
G	notCE	RnotW							PA2	PA4
H	ALE	GND	A6					Vin	GND	PA3
J	Wait	A5	A3	A1	DisInt ROM	Late	Read Data	Phase Up	Phase Down	PA1
K	A7	A4	A2	A0	Write Clock	not Write Gate	Early	not Write Data	not Index	PA0

Figure 4 .1 IMS M212 68 pin grid array

EUROPE

DENMARK

2730 HERLEV
Herlev Torv, 4
Tel. (45-42) 94.85.33
Telex: 35411
Telefax: (45-42) 948694

FINLAND

LOHJA SF-08150
Karjalankatu, 2
Tel. (358-12) 155.11
Telefax. (358-12) 155.66

FRANCE

94253 GENTILLY Cedex
7 - avenue Gallieni - BP. 93
Tel.: (33-1) 47.40.75.75
Telex: 632570 STMHQ
Telefax: (33-1) 47.40.79.10

67000 STRASBOURG
20, Place des Halles
Tel. (33) 88.75.50.66
Telefax: (33) 88.22.29.32

GERMANY

6000 FRANKFURT
Gutleutstrasse 322
Tel. (49-69) 237492-3
Telex: 176997 689
Telefax: (49-69) 231957
Teletex: 6997689=STVBP

8011 GRASBRUNN
Bretonischer Ring 4
Neukeferloh Technopark
Tel.: (49-89) 46006-0
Telex: 528211
Telefax: (49-89) 4605454
Teletex: 897107=STDISTR

3000 HANNOVER 51
Rotenburger Strasse 28A
Tel. (49-511) 615960
Telex 175118418
Teletex: 5118418 CSFBEH
Telefax: (49-511) 6151243

5202 HENNEF
Reuther Strasse 1A-C
Tel. (49-2242) 6088
(49-2242) 4019/4010
Telefax: (49-2242) 84181

8500 NÜRNBERG 20
Erlenstegenstrasse, 72
Tel.: (49-911) 59893-0
Telex: 626243
Telefax: (49-911) 5980701

7000 STUTTGART 31
Mittlerer Pfad 2-4
Tel. (49-711) 13968-0
Telex: 721718
Telefax: (49-711) 8661427

ITALY

20090 ASSAGO (MI)
V.le Milanofiori - Strada 4 - Palazzo A/4/A
Tel. (39-2) 89213.1 (10 linee)
Telex: 330131 - 330141 SGSAGR
Telefax: (39-2) 8250449

40033 CASALECCHIO DI RENO (BO)
Via R. Fucini, 12
Tel. (39-51) 591914
Telex: 512442
Telefax: (39-51) 591305

00161 ROMA
Via A. Torlonia, 15
Tel. (39-6) 8443341
Telex: 620653 SGSATE I
Telefax: (39-6) 8444474

NETHERLANDS

5652 AR EINDHOVEN
Meerenakkerweg 1
Tel.: (31-40) 550015
Telex: 51186
Telefax: (31-40) 528835

SPAIN

08021 BARCELONA
Calle Platon, 6 4th Floor, 5th Door
Tel. (34-3) 4143300-4143361
Telefax: (34-3) 2021461

28027 MADRID
Calle Albacete, 5
Tel. (34-1) 4051615
Telex: 46033 TCCEE
Telefax: (34-1) 4031134

SWEDEN

S-16421 KISTA
Borgarfjordsgatan, 13 - Box 1094
Tel.: (46-8) 7939220
Telex: 12078 THSWS
Telefax: (46-8) 7504950

SWITZERLAND

1218 GRAND-SACONNEX (GENEVA)
Chemin Francois-Lehmann, 18/A
Tel. (41-22) 7986462
Telex: 415493 STM CH
Telefax: (41-22) 7984869

UNITED KINGDOM and EIRE

MARLOW, BUCKS
Planar House, Parkway
Globe Park
Tel.: (44-628) 890800
Telex: 847458
Telefax: (44-628) 890391

AMERICAS

BRAZIL

05413 SÃO PAULO
R. Henrique Schaumann 286-CJ33
Tel. (55-11) 883-5455
Telex: (391)11-37988 "UMBR BR"
Telefax : (55-11) 282-2367

U.S.A.

NORTH & SOUTH AMERICAN
MARKETING HEADQUARTERS
1000 East Bell Road
Phoenix, AZ 85022
(1-602) 867-6100

SALES COVERAGE BY STATE

ALABAMA
Huntsville - (205) 533-5995

ARIZONA
Phoenix - (602) 867-6217

CALIFORNIA
Santa Ana - (714) 957-6018
San Jose - (408) 452-8585

COLORADO
Boulder (303) 449-9000

ILLINOIS
Schaumburg - (708) 517-1890

INDIANA
Kokomo - (317) 455-3500

MASSACHUSETTS
Lincoln - (617) 259-0300

MICHIGAN
Livonia - (313) 462-4030

NEW JERSEY
Voorhees - (609) 772-6222

NEW YORK
Poughkeepsie - (914) 454-8813

NORTH CAROLINA
Raleigh - (919) 787-6555

TEXAS
Carrollton - (214) 466-8844

FOR RF AND MICROWAVE
POWER TRANSISTORS CONTACT
THE FOLLOWING REGIONAL
OFFICE IN THE U.S.A.

PENNSYLVANIA
Montgomeryville - (215) 361-6400

ASIA / PACIFIC

AUSTRALIA

NSW 2027 EDGECLIFF
Suite 211, Edgecliff centre
203-233, New South Head Road
Tel. (61-2) 327.39.22
Telex: 071 126911 TCAUS
Telefax: (61-2) 327.61.76

HONG KONG

WANCHAI
22nd Floor - Hopewell centre
183 Queen's Road East
Tel. (852) 8615788
Telex: 60955 ESGIES HX
Telefax: (852) 8656589

INDIA

NEW DELHI 110001
LiasonOffice
62, Upper Ground Floor
World Trade Centre
Barakhamba Lane
Tel. (91-11) 3715191
Telex: 031-66816 STMI IN
Telefax: (91-11) 3715192

MALAYSIA

PULAU PINANG 10400
4th Floor - Suite 4-03
Bangunan FOP-123D Jalan Anson
Tel. (04) 379735
Telefax (04) 379816

KOREA

SEOUL 121
8th floor Shinwon Building
823-14, Yuksam-Dong
Kang-Nam-Gu
Tel. (82-2) 553-0399
Telex: SGSKOR K29998
Telefax: (82-2) 552-1051

SINGAPORE

SINGAPORE 2056
28 Ang Mo Kio - Industrial Park 2
Tel. (65) 4821411
Telex: RS 55201 ESGIES
Telefax: (65) 4820240

TAIWAN

TAIPEI
12th Floor
325, Section 1 Tun Hua South Road
Tel. (886-2) 755-4111
Telex: 10310 ESGIE TW
Telefax: (886-2) 755-4008

JAPAN

TOKYO 108
Nisseki - Takanawa Bld. 4F
2-18-10 Takanawa
Minato-Ku
Tel. (81-3) 3280-4121
Telefax: (81-3) 3280-4131