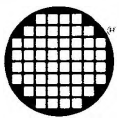
inmos

Product data



IMS T212 transputer

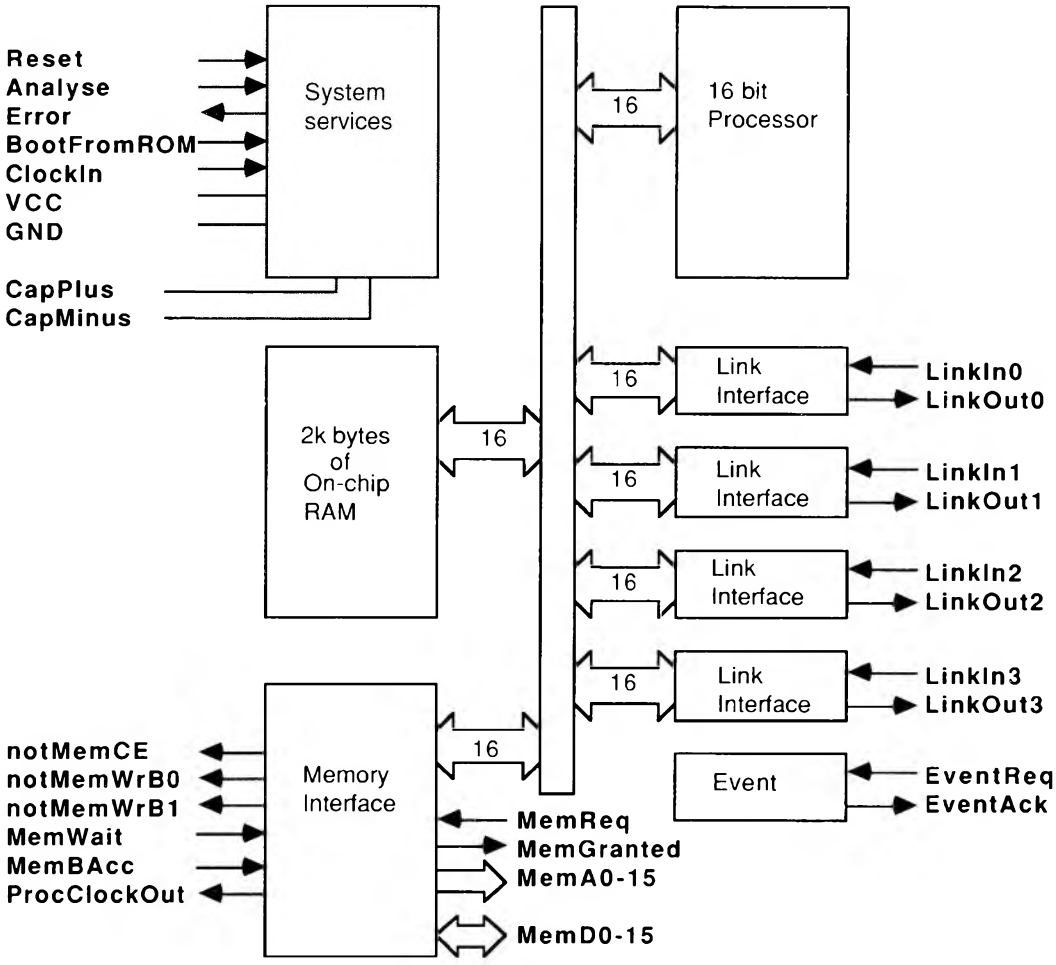
The IMS T212 is a 16-bit member of a family of transputers, all of which are consistent with the INMOS transputer architecture, described in the transputer architecture manual.

This manual details the product specific aspects of the IMS T212 and contains data relevant to the engineering and programming of the device.

Other information relevant to transputer products is contained in the occam programing manual (supplied with INMOS software products and available as a separate publication), and the transputer development system manual (supplied with the development system).

The examples given in this manual 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.

This edition of the manual is dated October 27, 1986.


Reset
Analyse
Error
BootFromROM
ClockIn
VCC
GND
CapPlus
CapMinus
System services
16
16 bit Processor
2k bytes of On-chip RAM
16
16
Link Interface
LinkIn0
LinkOut0
16
Link Interface
LinkIn1
LinkOut1
16
Link Interface
LinkIn2
LinkOut2
16
Link Interface
LinkIn3
LinkOut3
Event
EventReq
EventAck
notMemCE
notMemWrB0
notMemWrB1
MemWait
MemBAcc
ProcClockOut
Memory Interface
16
MemReq
MemGranted
MemA0-15
MemD0-15

IMS T212

The IMS T212 integrates a 16-bit microprocessor, four standard transputer communications links, 2K bytes of on-chip RAM, a memory interface and peripheral interfacing on a single chip, using a 1.5 micron CMOS process.

Processor

The design achieves compact programs, efficient high level language implementation and provides direct support for the occam model of concurrency. Procedure calls, process switching and interrupt latency are all sub-microsecond.

The processor shares its time between any number of concurrent processes. A process waiting for communication or a timer does not consume any processor time. Two levels of process priority enable fast interrupt response to be achieved.

Links

The T212 uses a DMA block transfer mechanism to transfer messages between memory and another transputer product via the INMOS links. The link interfaces and the processor all operate concurrently, allowing processing to continue while data is being transferred on all of the links.

Memory

The 2K bytes of static RAM provide a maximum data rate of 40 MBytes/sec with access for both the processor and links.

Memory interface

The T212 can directly access a linear address space up to 64 Kbytes, and has a 16-bit wide data bus and a 16-bit wide address bus, non-multiplexed, providing a data rate of up to 20 MBytes/sec, and supporting word or byte organisation. The data bus can be dynamically configured to be 16-bits or 8-bits wide.

Peripheral interface

The memory controller supports memory mapped peripherals, which may use DMA. Links may be interfaced to peripherals via an INMOS link adaptor. A peripheral can request attention via the event pin.

Time

The processor includes timers for both high and low priority processes.

Error handling

High-level language execution is made secure with array bounds checking, arithmetic overflow detection etc. A flag is set when an error is detected. The error can be handled internally by software or externally by sensing the error pin. System state is preserved for subsequent analysis.

The optimal method of programming the T212 processor is to use occam. See the occam programming manual for full details. Several standard languages are also supported. These include C, Fortran and Pascal.

The processor provides direct support for the occam model of concurrency and communication. It has a scheduler which enables any number of concurrent processes to be executed together, sharing the processor time. The number of registers which hold the process context is small and this, combined with fast on-chip RAM, provides a sub-microsecond process switch time.

Process communication is implemented by memory to memory block move operations. These fully utilize the bandwidth available from the on-chip RAM.

2.1 T212 types

The implementation of occam for the T212 transputer supports the following types:

CHAN OF *type* Each communication channel provides communication between two concurrent processes. Each channel allows the communication of data of the specified type.

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 $-32768 <= n < 32768$

INT16 Signed integers **n** in the range $-32768 <= n < 32768$

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

REAL64 Floating point numbers stored using a sign bit, 11 bit exponent and 52 bit fraction in ANSI/IEEE Standard 754-1985 representation

2.2 T212 process multiplexing

The T212 supports two levels of priority - in occam notation, a **PRI PAR** (priority parallel) process may have two components. The priority 1 (low priority) processes are executed whenever there are no active priority 0 (high priority) processes.

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.

Low priority processes

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 its 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.

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

To ensure that each low priority process proceeds, high priority processes should never occupy the processor continuously for a period of time equal to a timeslice period. A good guideline is to ensure that, if there are a total of **n** high priority processes, then each limits its activity to much less than 1/**n**'th of any 1 millisecond period.

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 (from when the channel becomes ready to when the process starts executing) is typically 19 processor cycles, maximum 53 cycles (assuming use of on-chip RAM).

2.3 T212 Error flag

Expressions which cause arithmetic overflow are invalid, and processes which cause array bound violations are invalid. If the compiler is unable to check that a given construct contains only valid expressions and processes, then extra instructions are compiled to perform the necessary checks at runtime. If the result of the check indicates that an invalid expression or invalid process has occurred, then the processor's **Error** flag is set.

In the implementation of occam, the offending process stops when the **Error** flag is set.

The T212 can be initialised so that the processor halts when the **Error** flag is set. The appropriate initialisation sequence is provided by the development system.

If the processor has been halted as the result of an error, the links continue with any outstanding transfers, and the transputer may be analysed. See section 3.2.

When a high priority process pre-empts a low priority process, the current value of the **Error** flag is saved; the **Error** flag itself is maintained. When, finally, there are no high priority processes able to run, the current state of the **Error** flag is lost, and the preserved state is restored as part of commencing to execute the pre-empted low priority process.

This ensures that the state of the **Error** flag is preserved during the evaluation of an expression or a sequence of assignments.

2.4 T212 memory map

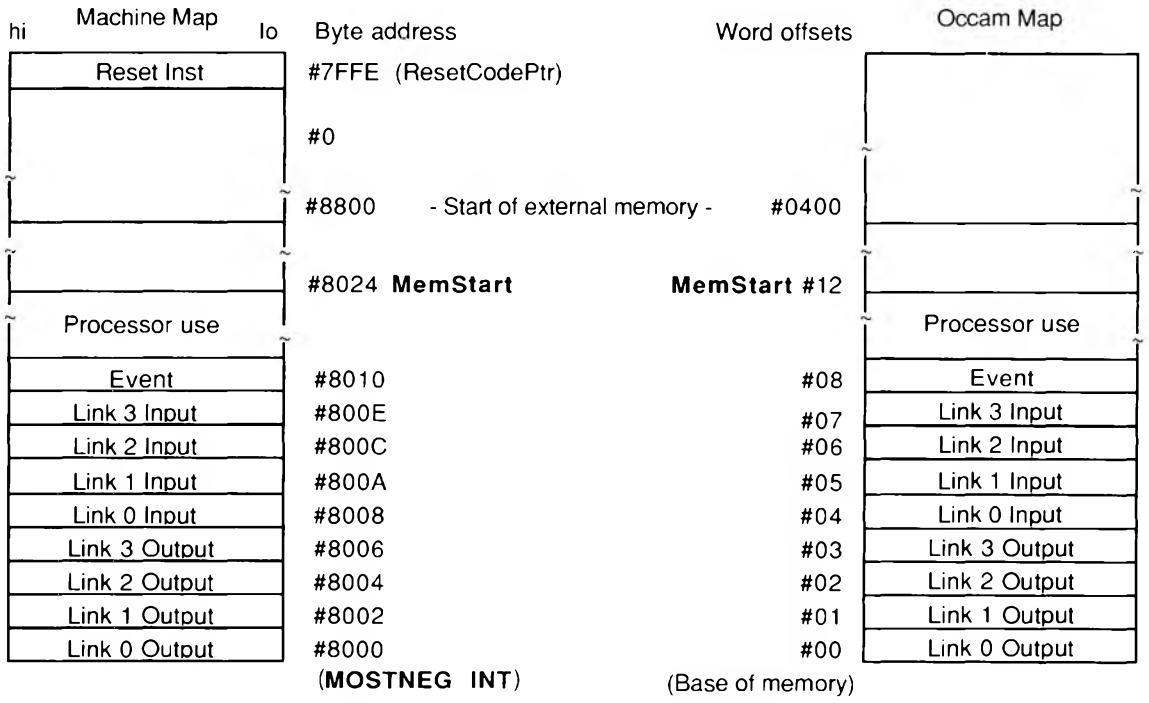
The address space of the T212 is signed and byte addressed. Words are aligned on two-byte boundaries. Addresses in the range #8000 to #87FF reference on-chip memory.

The first 18 words of the address space are used for system purposes. The next available location is, by convention, referred to as **MemStart**.

A suitable declaration of **MemStart**, for example is

```
VAL MemStart IS #12 :
```

The programmer can access locations in memory by using the occam mechanism of placement. This allows variables of any type to be placed at a location specified as a word offset from the base of memory. The placement of a byte array allows access to the byte components of a word. For example, a byte array could be placed in internal memory to optimize access, or a similar array could be placed in external memory to address one or more memory mapped devices.



This schema for calculating addresses is used to provide word length independent code, as though memory were an array of type **INT** (**[]INT**). The link addresses will always be found within the lower bounds of the memory array space.

The top of address space is used, by convention, for ROM based code. If the transputer is configured to bootstrap from ROM, then the processor commences execution from address #7FFE. Tools are supplied with the compiler and development system to enable the generation and placement of ROM images.

2.5 T212 timer

At the standard **ClockIn** cycle rate of 5MHz, the high priority timer has a resolution of one microsecond and cycles approximately every 65 milliseconds.

The low priority timer has a resolution of 64 microseconds. One second is exactly equal to 15625 ticks. Cycles approximately every four seconds.

2.6 T212 event pins

An attention-seeking peripheral may signal the T212 via the **EventReq** pin, which the T212 handshakes using **EventAck**. The T212 implements a hardware channel to allow a low to high transition on the **EventReq** pin to be communicated to a process as a synchronizing message.

An occam channel (which must already have been declared) may be associated with **EventReq** pin by a channel placement. The conventional name and the value used for this channel are given by

```
PLACE Event AT 8 :
```

Event behaves like an ordinary occam channel, and a process may synchronize with a low to high transition on the **EventReq** pin by using the occam construct:

```
Event ? signal
```

The process waits until the channel **Event** is ready. The channel is made ready by the transition on the **EventReq** pin (this may occur before the process attempts to input).

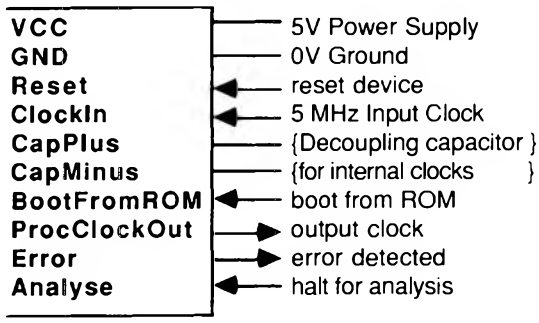
When the process is able to proceed, and if it executes at high priority, then it will take priority over any low priority process which may be executing when the transition occurs on the **EventReq** pin.

2.7 T212 link placement

The link addresses will always be found within the lower bounds of the memory array. The conventional names and the values used for these channels are

```
PLACE Link0Output AT 0 :
PLACE Link1Output AT 1 :
PLACE Link2Output AT 2 :
PLACE Link3Output AT 3 :
PLACE Link0Input  AT 4 :
PLACE Link1Input  AT 5 :
PLACE Link2Input  AT 6 :
PLACE Link3Input  AT 7 :
```

The system services comprise the clocks, power and initialisation used by the whole of the transputer. The **Reset** and **Analyse** inputs enable the T212 to be initialised, for example on power up, or halted in a way which preserves its state for subsequent analysis. Whilst the T212 is running both **Reset** and **Analyse** are held low. The **Error** signal is directly connected to the processor's **Error** flag. See section 2.3.



3.1 Reset

The T212 is initialised by pulsing **Reset** high whilst holding **Analyse** low. Operation ceases immediately and all state information is lost.

The processor then bootstraps. If the **BootFromROM** input is high it will start to execute code starting from address #7FFE. If **BootFromROM** is low it will bootstrap from a link, that is, it will load a program from a link and then execute it.

When initialising following power-on, a time is specified during which **VCC** must be within specification, **Reset** must be high, and the input on **ClockIn** must be oscillating. **Reset** is taken low after the specified time has elapsed.

During power on reset all link inputs must be held low. (N.B. all link outputs are made low by reset.)

3.2 Analyse

A system built from transputers may be brought to a halt in a consistent state which is preserved for subsequent analysis. This analysis is performed in a manner similar to bootstrapping. The transputer development system includes appropriate bootstrap and analysis software.

A signal may be applied to the **Analyse** inputs of all the transputers in the system. The system is analysed by first taking **Analyse** high. This causes each transputer to halt, after a short period of time, in a consistent internal state. When the system has halted, **Reset** is taken high for the specified hold time, after which it is taken low. **Analyse** is then taken low, at which time each transputer bootstraps.

When **Analyse** is taken high, the processor will halt within three timeslice periods (approximately three milliseconds), plus the time taken for any high priority process to cease processing. Any outputting links continue until they complete the remainder of the current word. Input links will continue to receive data. Provided that there are no delays in sending acknowledgements, the links in a system will therefore cease activity within a few microseconds. Sufficient time must be allowed to allow the processor to halt and link traffic to cease before **Reset** is asserted.

The system must be designed so that links connected to the external world are quiet during reset.

3.3 Bootstrapping and analysis of a "failed" system

The transputer has two methods of bootstrapping. Firstly, the conventional bootstrap which occurs after **Reset** and secondly, an analysis bootstrap which occurs after **Reset** plus **Analyse**. This second method allows the state of a system, which could well be a complex network of transputers, to be examined. For example, memory continues to be refreshed so that data is not lost. In both cases the mechanism used is very similar and the bootstrap or analysis code may be provided from either the external memory of the transputer, often in ROM, or if the transputer is a part of a network, via its communication links.

3.3.1 Bootstrapping

It is possible to bootstrap the T212 either by executing code held in ROM or by executing code received on a link. In addition, prior to bootstrapping from a link, it is possible to read or write to any memory location in a transputer's memory map via a link.

3.3.2 Bootstrapping from ROM

To bootstrap from ROM, the **BootFromROM** input is wired to **VCC**.

The T212 bootstraps from ROM by executing a process at low priority. Control is transferred to the top two bytes in memory (at #7FFE), which will invariably contain a backward jump into ROM. **`Memstart`** (#8024) is used as the location of the process workspace.

3.3.3 Bootstrapping from a link

To bootstrap from a link, the **BootFromROM** input is wired to **GND**.

The T212 bootstraps from a link by waiting for the first byte (the control byte) to arrive on any of the four link inputs.

If the value of the control byte is two, or greater, it is considered to be a count of the number of bytes to be input. The following bytes are then placed into memory at **`MemStart`** (#8024) and the T212 begins to execute the input code as a process at low priority by transferring control to **`MemStart`**. The memory space immediately above the loaded code is used as the process workspace.

The mechanism of bootstrapping from any link allows a network of transputers to be bootstrapped without the need for ROM or any external memory.

3.3.4 Peeking and Poking

A unique feature of the transputer allows any location of transputer memory, be it internal or external, to be read or written to from a link. This allows, with appropriate software, an external memory system to be debugged without the need to run a program on the system under development.

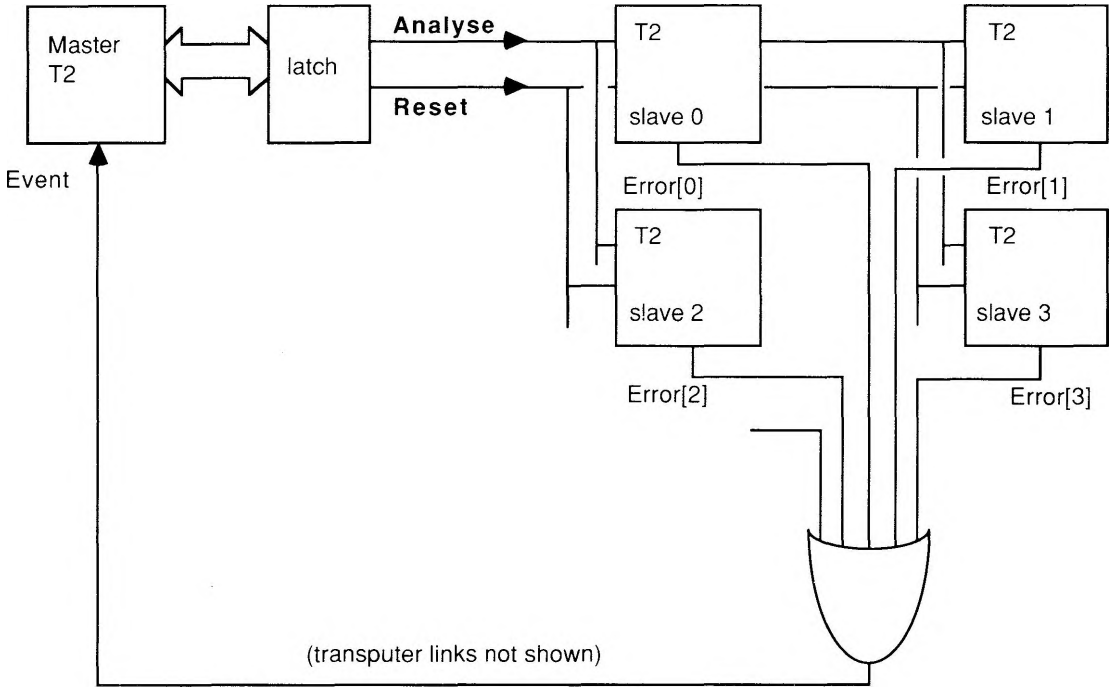
If, whilst the transputer is waiting to boot from a link, it receives a zero byte then a word address is input, followed by a word of data which is written to that address. The transputer then returns to the state of awaiting a message from any link.

If the first byte received is one, then a word address is input, a word of data is read from that address and is output down the corresponding output link. The transputer will then return to the state of awaiting a message from any link.

3.4 Using Error and Analyse

Analyse may be used in conjunction with the **Error** output to isolate errors in a multi-transputer system. A transputer which has signalled an error may be halted and subsequently analysed under the control of a 'master' transputer, using the methods described above. Or this could be achieved by connecting the 'OR' of all the **Error** pins to the **EventReq** pin on the master transputer, and connecting the **Analyse** and **Reset** pins to the master transputer as a 'peripheral'.

Error treatment in multi-transputer system

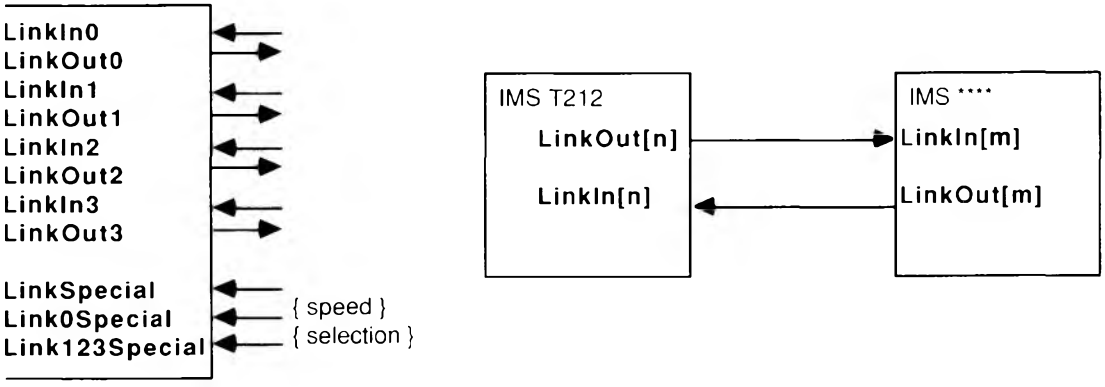


4.1 Standard transputer links

The T212 provides four standard links, each providing two uni-directional point-to-point occam channels.

The links implement the standard inter transputer communications protocol. The links are connected by wiring a **LinkOut[n]** to a **LinkIn[m]** and **LinkIn[n]** to a **LinkOut[m]**. Unused link inputs should be held to ground, **GND**.

Link connection



The T212 links wait until each full byte has been received before outputting the corresponding acknowledge packet. An acknowledge packet can be received at any time following the transmission of a data packet start bit.

At a link speed of 10Mbits/sec, data is transmitted at about 400Kbytes/sec in each direction, and the combined (bidirectional) data rate when the link carries data in both directions at once is 800Kbytes/sec.

4.1.1 Link speed selection

Link speed selection depends on the settings of three pins, **Link0Special**, **Link123Special** and **LinkSpecial**. These are shown in the tables below, a 0 indicating the signal should be held to **GND**, and a 1 indicating the signal should be held to **VCC**.

Table 1. Speed settings for Link 0.

LinkSpecial	**Link0Special**	**Speed at 5MHz** Mbits/sec	**Unidirectional data rate** Kbytes/sec	**Bidirectional data rate** Kbytes/sec
0	0	10	400	800
0	1	5	200	400
1	0	10	400	800
1	1	20	800	1600

Table 2. Speed settings for Links 1,2 and 3.

LinkSpecial	**Link123Special**	**Speed** Mbits/sec	**Unidirectional data rate** Kbytes/sec	**Bidirectional data rate** Kbytes/sec
0	0	10	400	800
0	1	5	200	400
1	0	10	400	800
1	1	20	800	1600

The data rates are reduced when performing transfers using slow external memory.

The memory interface has a 16-bit address bus and a 16-bit data bus. Word reads and writes are performed in two processor cycles, a processor cycle being defined as one cycle of **ProcClockOut**.

Wait states can be introduced by taking **MemWait** high. This signal is sampled during the first processor cycle of the memory access. If it is found to be high, processor cycles are added to the memory access until subsequent sampling detects **MemWait** to be low, at which time the memory access proceeds normally.

The T212 memory interface will by default perform word access at even memory locations. Byte accessing can be achieved by taking **MemBAcc** high. The state of this signal is latched during the second half of the first cycle. The first byte is accessed at the word address, and the second byte is accessed at the word address + 1 at which time **MemA0** is high.

Two write enable signals control which byte of data is to be written by the memory interface. These are active low and are called **notMemWrB0** and **notMemWrB1**.

The active low signal **notMemCE** is used to enable external memory on both read and write cycles. External memory cycles are divided into 4 T-states, **T1** to **T4**. Each T-state can be defined as follows:

T1 Address and Write Enables set-up period.
T2 Data set-up for writes.
T3 Access time extension for slow RAMs.
T4 Bus turnround, end of cycle.

Note: When internal memory is accessed **MemA0-15** takes the value of the internal address and **notMemWrB0-1** takes the value of the interupt write enables. The signal **notMemCE** however is not asserted.

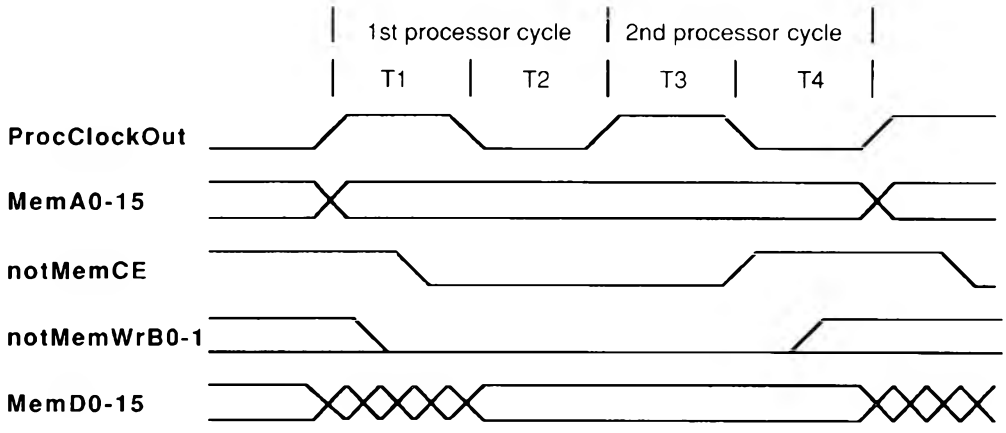
5.1 Word access

This is the default mode and also the more efficient. The processor uses **notMemWrB0** and **notMemWrB1** on write cycles to select which bytes are to be written.

Write cycles

The processor supports early write cycles in both word and byte accessing modes. Write enables are asserted before the chip enable signal **notMemCE** is set low. This reduces memory access time and therefore the risk of bus contention.

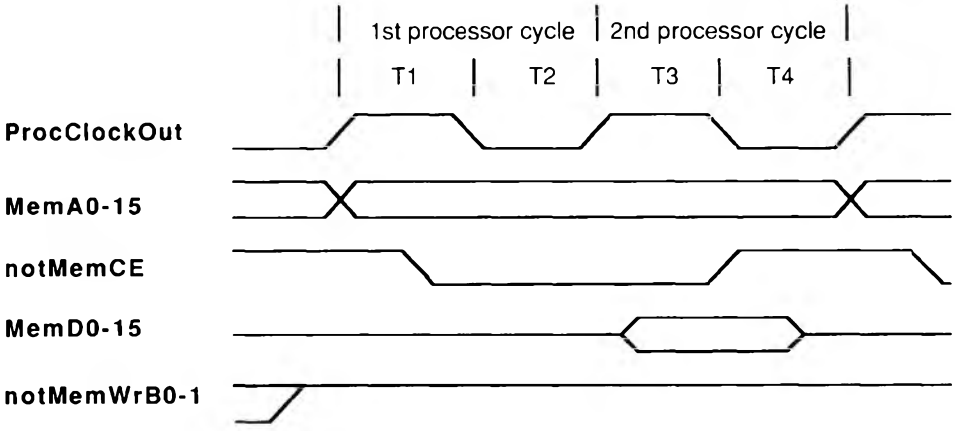
Word Write Cycle



Read cycles

During this cycle the data bus is allowed to float from the time that a write enable is deasserted until the memory system or peripherals drive the data bus. Memory data is latched at the end of **T3** and must be held until **notMemCE** has gone high.

Word Read Cycle



5.2 Byte access

Byte access mode uses the **MemD0-D7** pins to access byte data buses.

The processor performs two cycles for each read during byte access. The least significant byte of a word is expected on **MemD0-D7** on the first cycle and the most significant byte is expected on **MemD0-D7** on the second cycle.

Similarly, the processor performs two cycles for each write during byte access. The least significant byte of a word is placed on **MemD0-D7** on the first cycle and the most significant byte is placed on **MemD0-D7** on the second cycle.

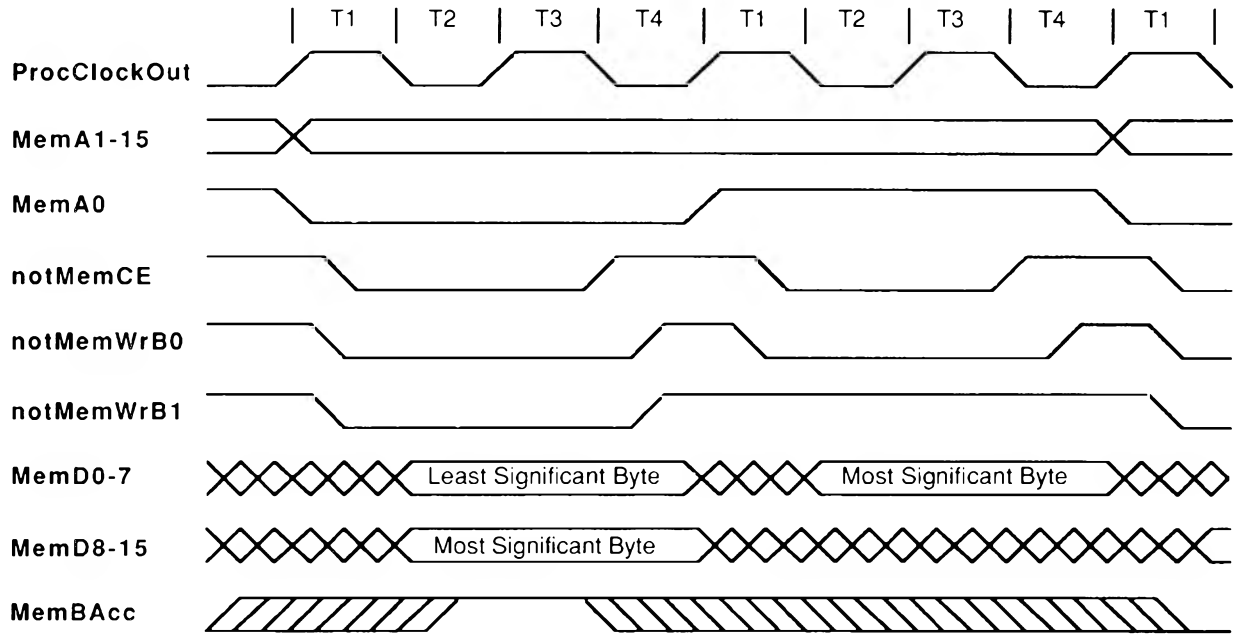
When the processor accesses a single byte, the interface still performs two cycles, one of which is a dummy cycle.

To enable byte access, **MemBAcc** must be taken high before the end of **T2**. **MemBAcc** must be asserted before **T3** of the first byte access, and can be deasserted at the beginning of **T4** on the second byte access.

As the processor still expects 16-bit data internally, the memory interface deals with the task of assembling 2 bytes into a word and vice versa, and asserting the appropriate control strobes. This means that in byte access mode two memory accesses are performed each time the memory interface reads or writes data.

MemA0 is low for the least significant byte access and high for the most significant byte access.

Write Cycle in Byte Access Mode



5.3 The use of MemWait

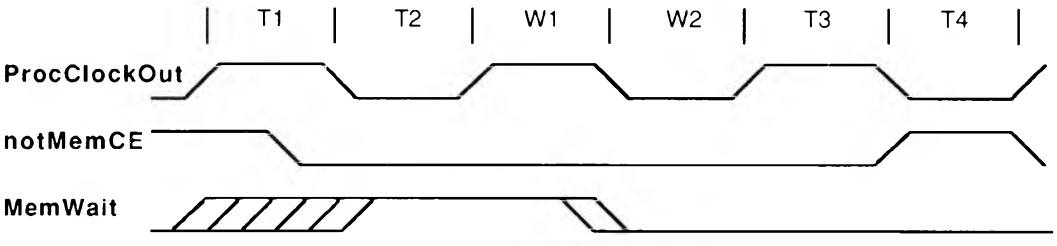
Memory cycle times can be extended by asserting **MemWait** at the correct time.

MemWait is sampled during **T2** and so must be set up before this state occurs.

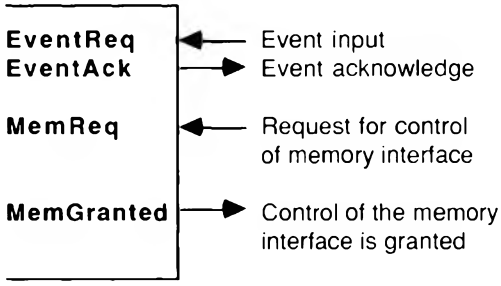
The memory interface of the T212 has been designed to interface to a simple wait state generator such as a digital delay line.

When **MemWait** is held high, the memory interface will add a further two Tstates. Further cycles will be added as long as **MemWait** is held high during **T2**.

Single Wait State Timing

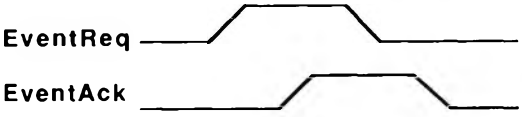


Event Request block diagram



The two event signals, **EventReq** and **EventAck**, together provide a handshaken interface with an occam process executing in the processor.

Event request signal protocol



External logic takes **EventReq** high when the logic wishes to communicate with a process in the transputer. The rising edge of **EventReq** makes an external channel ready to communicate with the process. (This channel is additional to the external channels of the links.) When both the channel is ready and a process is ready to input from the channel, then the processor takes **EventAck** high and the process is scheduled. At any time after this point the external logic may take **EventReq** low, following which the processor will set **EventAck** low. After **EventAck** goes low, **EventReq** may go high to indicate the next event. Any further communication or synchronization necessary (for example, to tell external logic that the process has acted in response to the event) must be programmed explicitly.

If the process has high priority, and there is no other high priority process already running, then the maximum latency is 53 processor cycles, assuming that all memory accesses are to on-chip RAM. The typical latency is 19 processor cycles.

EventReq should be held low on reset.

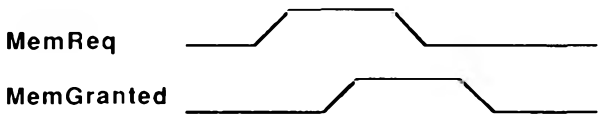
DMA via the memory interface

Peripheral controllers may access the whole of the external memory address space for DMA transfers using the asynchronous signals **MemReq** and **MemGranted**, which provide a handshake protocol for requesting and acknowledging control of the external address and data buses.

If **MemReq** is taken high the processor will finish the current memory cycle before allowing a peripheral to take control of the external data and address buses. In byte access mode, this means that control will not be given until the 2nd cycle has been performed.

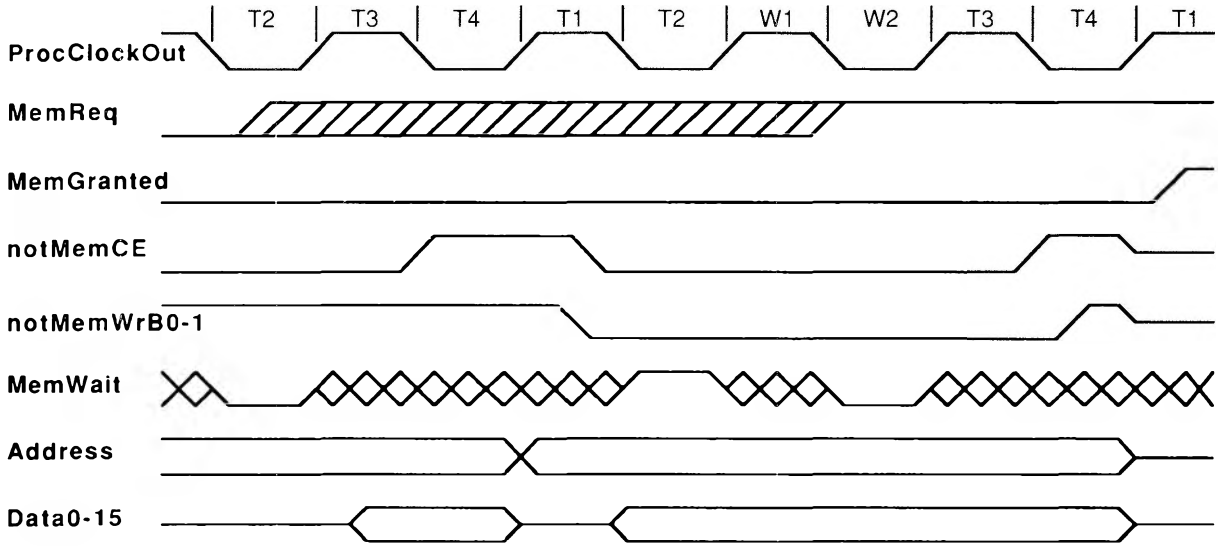
When the memory interface has completed its cycle, the T212 tristates the bus and asserts **MemGranted**. The peripheral cannot use the interface until it has received **MemGranted**. When the peripheral device deasserts **MemReq**, the processor releases **MemGranted** and starts memory access.

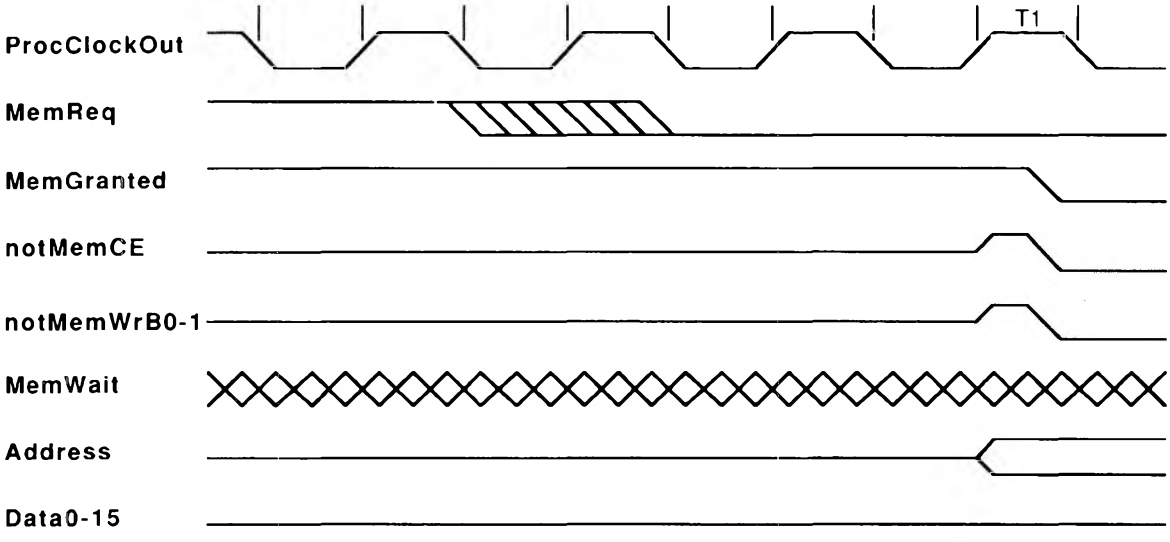
Memory Request Handshake



The processor and links can still continue to operate to internal memory while DMA proceeds to external memory, provided the processor or links do not try to use the external memory during DMA. The acknowledge signal **MemGranted** will function regardless of the state of **Reset** and **Analyse**.

Memory Request including Wait State





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.

7.1 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 following abbreviations are used to represent the quantities indicated.

np number of component processes
ne number of processes earlier in queue
r 1 if INT 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 multiplier is 0)

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** 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 *type*	3.1	3.1
[size]CHAN OF *type*	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	23
/	2	24
REM	2	22
>>, **<<**	2	3+**p**
Modulo Arithmetic operators		
PLUS	2	2
MINUS	1	1
TIMES(fast multiply)	1	4+**Tb**

	Size (bytes)	**Time** (cycles)
Boolean operators		
OR	4	8
AND, **NOT**	1	2
Comparison operators		
= constant	0	1
= variable	2	3
<> constant	1	3
<> 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 call		
	3.5+(nparams-2)*1.1 +nvecparams*2.3	16.5+(nparams-2)*1.1 +nvecparams*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}

Figures in curly brackets 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.

7.1.1 Fast multiply, TIMES

The T212 has a fast multiplication instruction ('product'). If **Tb** is the position of the most significant bit set in the multiplier, then the time taken for a fast multiply is 4+**Tb**. The time taken for a multiplication by zero is 3 cycles. For example, if the multiplier is 1 the time taken is 4 cycles, if the multiplier is -1 (all bits set) the time taken is 19 cycles. Implementations of occam on the transputer take advantage of this instruction. The modulo operator **TIMES** is mapped onto the instruction and is treated as non-commutative, the expression on the right of the expression being the multiplier.

The fast muliplication instruction is also used in occam implementations for the multiplication implicit in multi-dimensional array access.

7.1.2 T212 arithmetic procedures

A set of predefined procedures (**PROC**s) are provided to support the efficient implementation of multiple length and floating point arithmetic. In the following table, **n** gives the number of places shifted and all parameters are assumed to be local. Full details of these procedures are provided in the occam reference manual supplied as part of the development system and available as a separate publication.

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

PROC		Cycles	+Cycles for Parameter Access (Assuming local variables)
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>=16)	n-11	8
SHIFTLEFT	(n<16)	4+n	8
	(n>=16)	n-11	8
NORMALISE	(n<16)	n+6	7
	(n>=16)	n-9	7
	(n=32)	4	7
ASHIFTRIGHT		SHIFTRIGHT+2	5
ASHIFTLEFT		SHIFTLEFT+4	5
ROTATERIGHT		SHIFTRIGHT	7
ROTATELEFT		SHIFTLEFT	7

7.1.3 Floating point operations

Floating point operations are provided by a run-time package, which requires approximately 2000 bytes of memory for the double length arithmetic operations, and 2500 bytes for the quadruple length arithmetic operations. The following table summarizes the estimated performance of the package.

		Processor cycles (typical)	Processor cycles (worst)
REAL32	+, -	530	705
	*	650	705
	/	1000	1410
	<, >, =, >=, <=, <>	60	60
REAL64	+, -	875	1190
	*	1490	1950
	/	2355	3255
	<, >, =, >=, <=, <>	60	60

7.1.4 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**. The value of **e** is 1 for no wait states.

The number of additional cycles required to access data in external memory is **e**. If 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 (2**e** - 1)/4 per byte of program. A transfer of control may be estimated as requiring **e** + 3 cycles.

These estimates may be refined for various constructs. In the following table, **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**-1	0
IF	3**en**-1	**en**
Replicated **IF**	6**en**+9e-12	(5**e**-2)**n**+6
Replicated **SEQ**	(4**e**-3)**n**+3e	(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**)

The effective rate of INMOS links is slowed down on output from external memory; by **e** cycles per word output, and on input to external memory at 10 Mbits/sec by **e**-6 cycles per word if **e**≥6.

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 root computation.

	e	1	2	3	4	**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)	1.1	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

7.2 T212 speed selections

The following table illustrates the designation of the T212 speed selections.

Designation	**Instruction throughput**	**Processor clock speed**	**Processor cycle time**	**Input clock frequency**
IMS T212-17	8.75 MIPS	17.5 MHz	57 ns	5 MHz
IMS T212-20	10 MIPS	20 MHz	50 ns	5 MHz

Parameters given in this section will be revised as a result of fuller characterization.

8.1 Absolute maximum ratings

Parameter		Min	Max	Unit	Note
VCC	DC supply voltage	0	7.0	V	1, 2, 3
VI,VO	Input or output voltage on any pin	-0.5	**VCC**+0.5	V	1, 2, 3
OSCT	Output short circuit time (one pin)		1	S	1
TS	Storage temperature	-65	150	°C	1
TA	Ambient temperature under bias	-55	125	°C	1
PD	Power dissipation rating		1	W	

Notes

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

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

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electric 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. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level such as **GND**.

8.2 Recommended operating conditions

Parameter		Min	Max	Unit	Note
VCC	DC supply voltage	4.5	5.5	V	1
VI,VO	Input or output voltage	0	**VCC**	V	1,2
II	Input current		±25	mA	3
CL	Load capacitance on any **MemAD** pin		50	pF	
TA	Operating temperature range	0	70	°C	

Notes

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

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

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

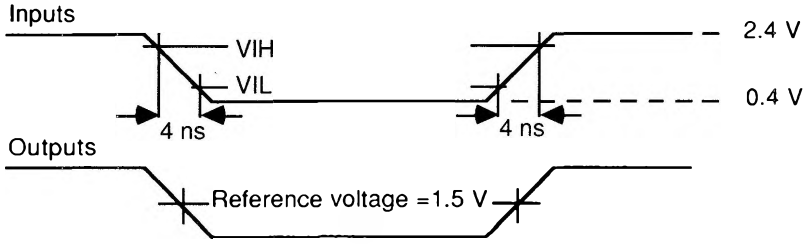
8.3 DC characteristics

Parameter		Conditions	Min	Max	Unit
VIH	High level input voltage		2.0	**VCC**+0.5	V
VIL	Low level input voltage		-0.5	0.8	V
II	Input current	**GND**<**VI**<**VCC**		±10	μA
VOH	Output high voltage	**IOH**=2mA	**VCC**-1		V
VOL	Output low voltage	**IOL**=4mA		0.4	V
IOS	Output short circuit current	**GND**<**VO**<**VCC**		50	mA
IOZ	Tristate output current	**GND**<**VI**<**VCC**		±10	μA
PD	Power dissipation			0.7	W
CIN	Input capacitance	f=1 MHz		7	pF
COZ	Output capacitance tristate	f=1 MHz		10	pF

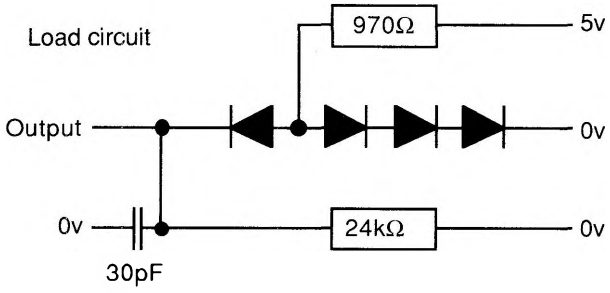
Note :
4.5 V < **VCC** < 5.5 V
0 °C < **TA** < 70 °C
input clock frequency = 5 MHz
All voltages are with respect to **GND**

8.4 Measurement of AC characteristics

Reference points for AC measurements



Load circuit for AC measurements



The load circuit approximates to two Schottky TTL loads, with total capacitance of 30 pF.

8.5 Connection of INMOS serial links

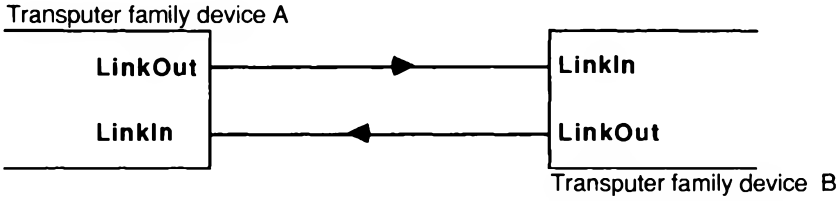
INMOS serial links can be connected in 3 different ways depending on their environment:

1 Directly connected

2 Connected via a series matching resistor

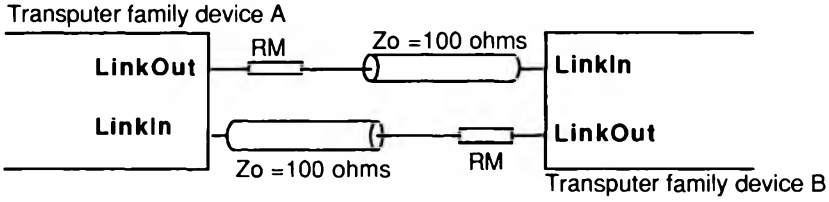
3 Connected via buffers

Direct connection



Direct connection is suitable for short distances on a printed circuit board.

Matched line

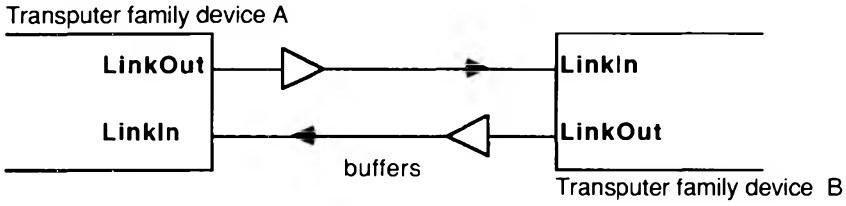


For long wires, approximately >30 cm, then a 100ohm transmission line should be used with series matching resistors.

Parameter		Nom	Max	Unit
RM	Series matching resistor for 100 ohm line.	47		ohm
TD	Delay down line		0.4	bit time

Note that if two connected devices have different values for **TD**, the lower value should be used. With series termination at **LinkOut** the transmission line reflection must return within 1 Bit time. Otherwise line buffers should be used.

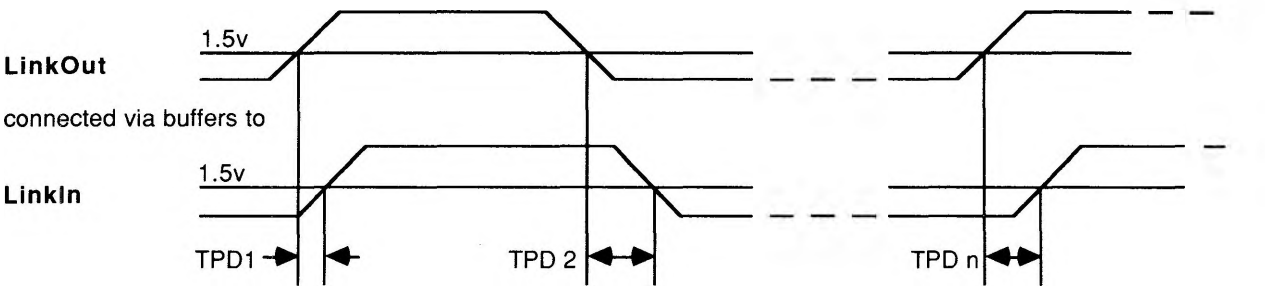
Buffered links



If buffers are used their overall propagation delay, TPD, should be stable within the skew tolerance.

Parameter	Max	Unit
Skew in buffering at 5 Mbits/sec	30	ns
Skew in buffering at 10 Mbits/sec	10	ns
Skew in buffering at 20 Mbits/sec		ns
Rise and fall time of **LinkIn** (10% to 90%)	20	ns

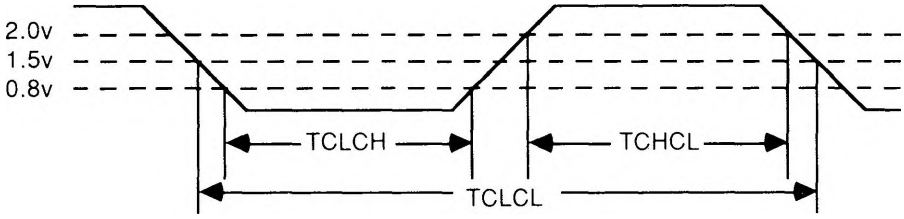
The above figures indicate that buffered links can be realised at 5Mbits/sec and 10Mbits/sec. For the case of 20 Mbits/sec, the maximum value can only be specified as a result of further characterisation.



The absolute value of TPD is immaterial because data reception is asynchronous. However, TPD will vary from moment to moment because of ground noise, variation in the power supplies of buffers and the difference in the delay for rising and falling edges. This will vary the length of data bits reaching **LinkIn**. *Skew* is the difference between the maximum and minimum instantaneous values of TPD.

8.6 AC characteristics of system services

ClockIn waveform



Parameter		Min	Nom	Max	Unit	Note
TCHCL	Clock pulse width high	40			ns	1
TCLCH	Clock pulse width low	40			ns	1
	Rise and fall time of **ClockIn** (10% to 90%)			10	ns	1
TCLCL	Clock period		200		ns	2
	Difference in frequencies of **ClockIn** for two devices connected by a link			400	ppm	6
Cap	Decoupling capacitor between **CapPlus** and **CapMinus**	1			μF	3
TRHRL	**Reset** pulse width high	8			**ClockIn** periods	4
	Time **VCC** must be valid and **Clockin** running before **Reset** goes low	10			ms	5
	BootFromROM setup time before **Reset** or **Analyse** goes low	0				
	BootFromROM hold time after **Reset** goes low	50			ms	
	Analyse pulse width	8			**Clockin** periods	

Notes

1 The clock transitions must be monotonic within the range between **VIH** and **VIL**.

2 The **TCLCL** parameter is measured between corresponding points on consecutive falling edges.

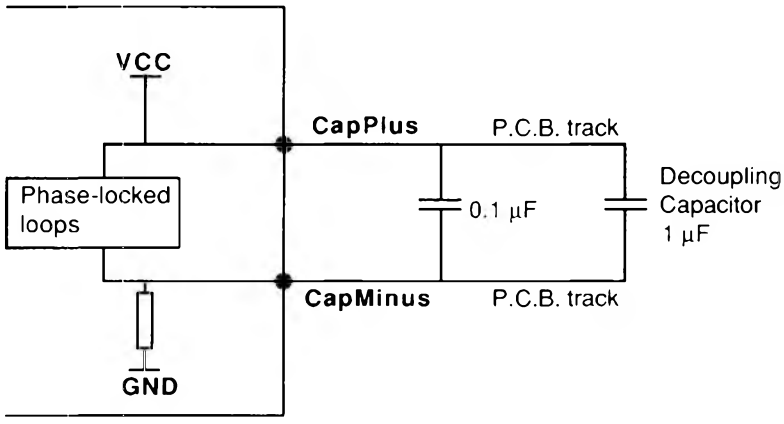
3 A 1 μF tantalum or ceramic low inductance, low leakage capacitor must be connected between **CapPlus** and **CapMinus** to decouple the supply to the on-chip clock generator. An additional good quality 0.1 μF ceramic capacitor should be connected in parallel with this. PCB track lengths to the capacitor should be minimised. No power supply should flow in these tracks.

4 Link inputs must be held low during reset. Reset forces link outputs low.

5 At power on reset.

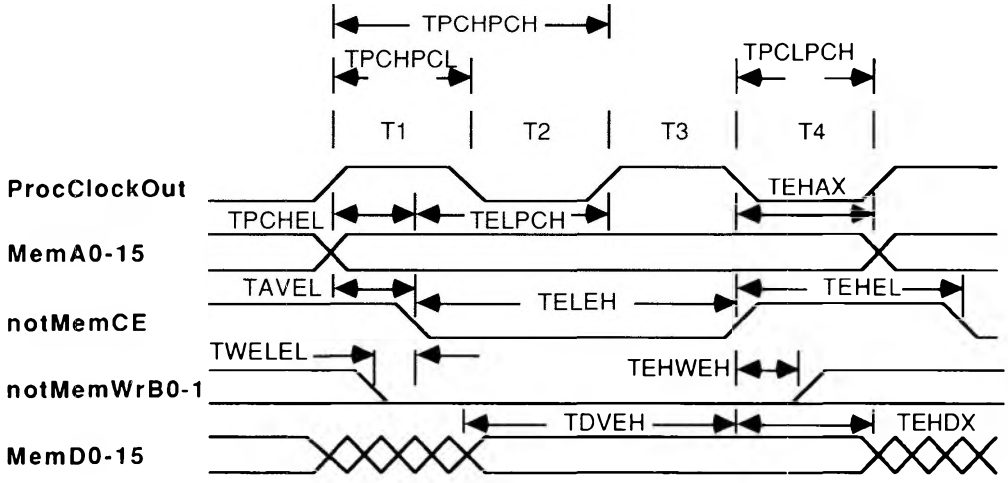
6 This value allows the use of low cost 200ppm crystal oscillators.

Recommended PLL Decoupling.

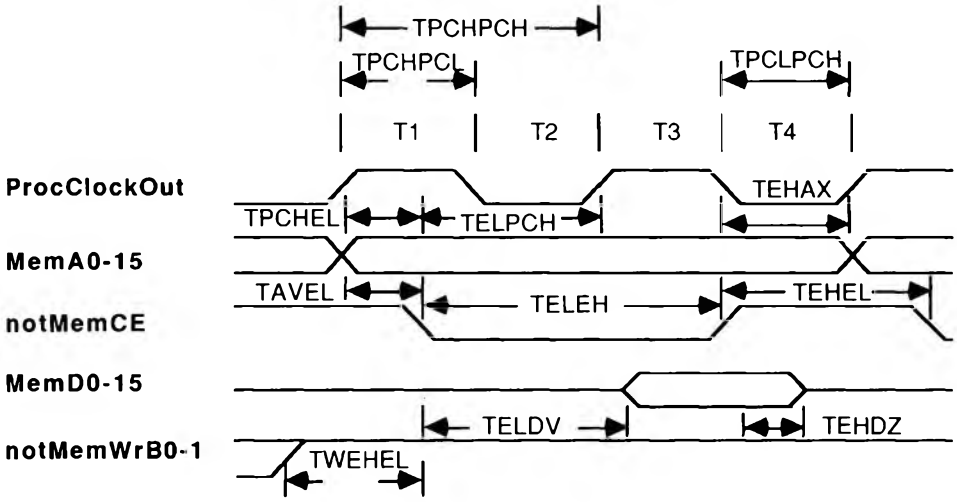


8.7 Memory interface AC characteristics

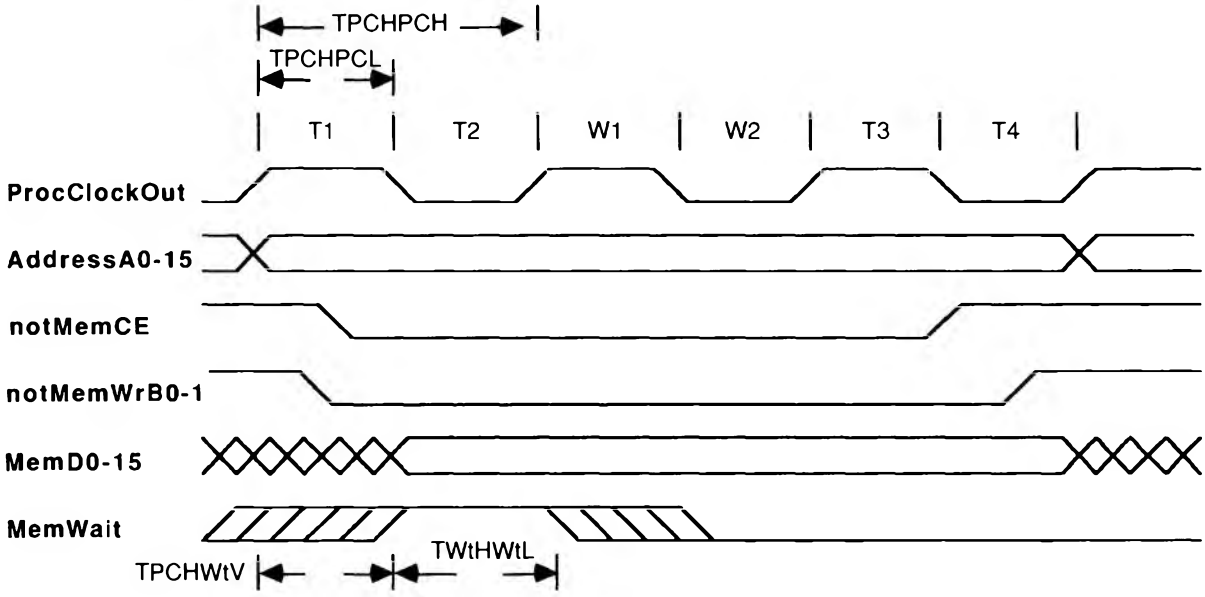
Word Write Cycle A.C timing.



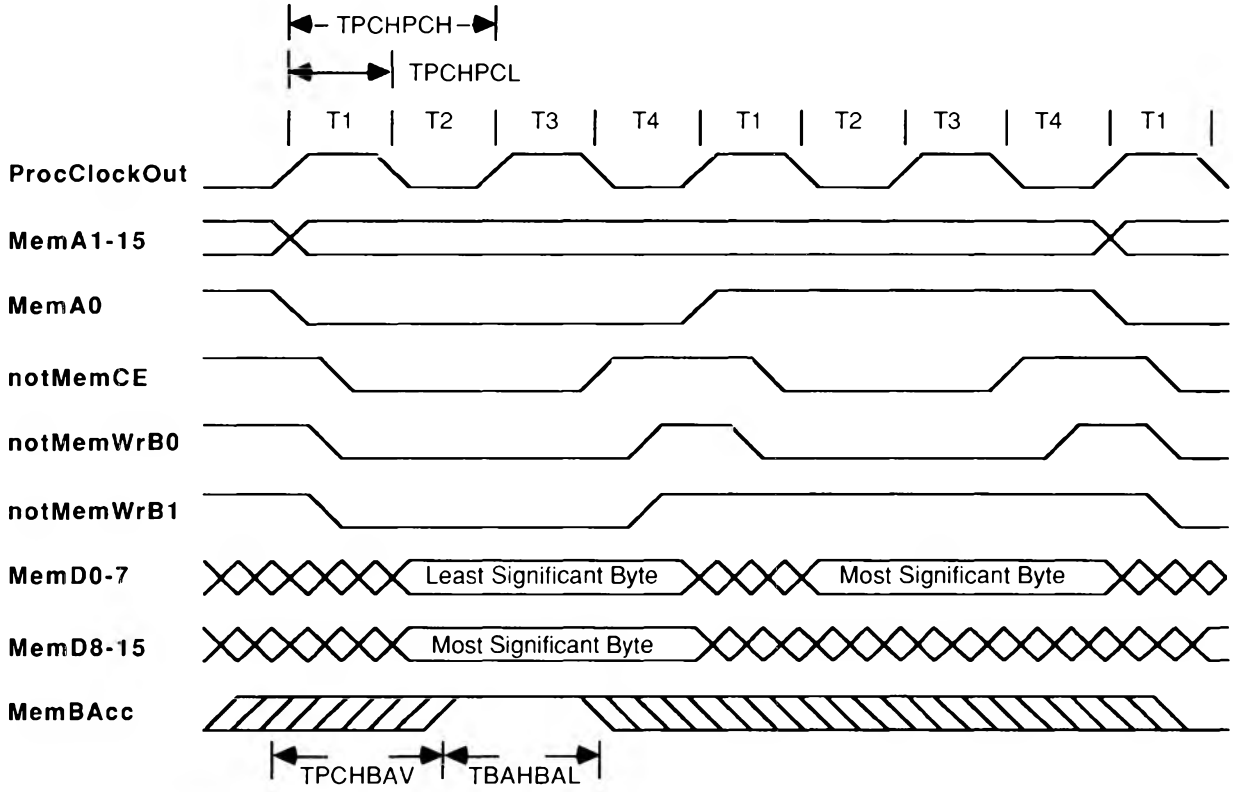
Read Cycle A.C timing.



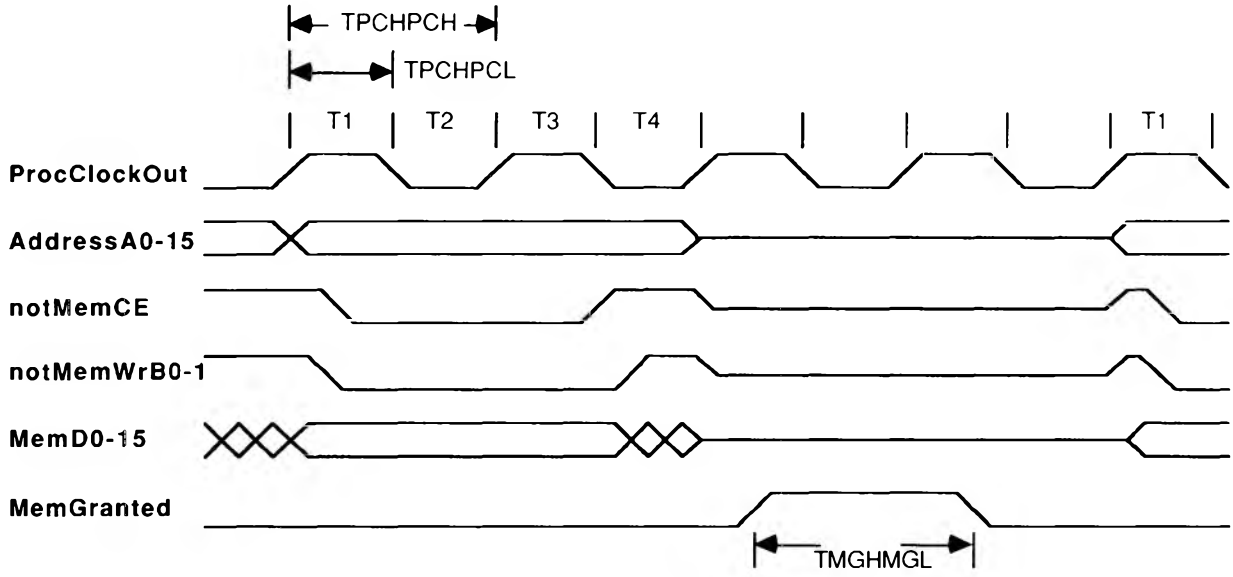
A.C timing for a single Wait State during a Write Cycle.



A.C timing to assert Byte Access.



A.C timing to assert Memory Request.



The AC characteristics of the memory interface are dependent upon the speed of the transputer and the use of wait states.

Note

The parameter table can be used to calculate figures for all T212 speed selections.

n represents the value used in the internal divide network to provide frequency multiplication. Processors therefore with the following cycle times have a corresponding division factor **n** of the period of **ClockIn** (200 ns) as follows:

ProcClockOut Period	n
44 ns	4.5
50 ns	4.0
57 ns	3.5

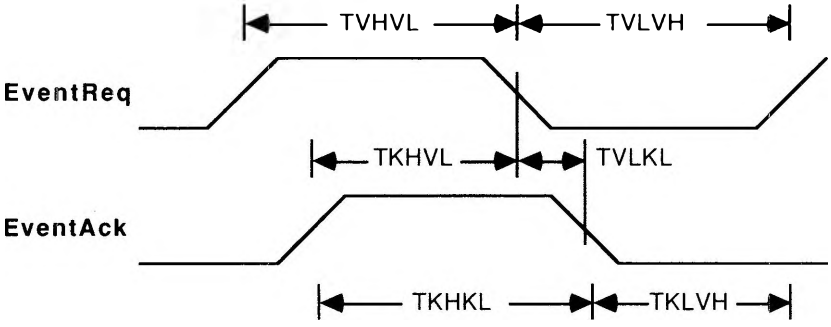
Parameters given in this section will be revised as a result of fuller characterization.

Parameter		Minimum	Maximum	Unit
TPCHPCH	**ProcClockOut** period	200/**n**	-	ns
TPCHPCL	**ProcClockOut** pulse width high	(**TPCHPCH**/2)-1	-	ns
TPCLPCH	**ProcClockOut** pulse width low	**TPCHPCH**-**TPCHPCL**	-	ns
TPCHEL	**ProcClockOut** rising edge to Chip Enable low	1	-	ns
TELPCH	Chip Enable low to end of first cycle	**TPCHPCH**-1	-	ns
TEHAX	Address hold time	**TPCHPCH**/2	-	ns
TAVEL	Address setup time	**TPCHPCH**/4	-	ns
TELEH	Chip Enable low time	5*__TPCHPCH__/4	-	ns
TEHEL	Chip Enable high time	3*__TPCHPCH__/4	-	ns
TWELEL	Write command setup time	3	-	
TEHWEH	Write command hold time	**TPCHPCH**/2	-	ns
TDVEH	Data setup time	**TPCHPCH**	-	ns
TEHDX	Data hold time	**TPCLPCH**	-	ns
TELDV	Chip Enable access time	-	(5*__TPCHPCH__/4)-23	ns *
TEHDZ	Read data hold time	0	-	ns
TWEHEL	Write command deassert	0	-	ns
TPCHWtV	**ProcClockOut** rising edge to Wait sampled	-	(3*__TPCHPCH__/4)-23	ns *
TWtHWtL	Wait hold time per wait state	10	-	ns
TPCHBAV	**ProcClockOut** rising edge to ByteAccess valid	-	(3*__TPCHPCH__/4)-23	ns *
TBAHBAL	ByteAccess hold time	10	-	ns
TMGHMGL	**MemGranted** pulse width high	**TPCHPCH**	-	ns

* : This parameter includes a constant 23ns delay, which consists of 16ns delay through the output pad and 7ns delay through the input pad.

8.8 Peripheral interfacing AC characteristics

Event signals waveforms



Parameter		Min	Max	Unit
TVHVL	**EventReq** pulse width high	2		processor cycles
TVLVH	**EventReq** pulse width low	2		processor cycles
TVLKL	Falling edge delay from **EventReq** to **EventAck**	0	2	processor cycles
TKHKL	**EventAck** pulse width high	2		processor cycles
TKHVL	Delay from **EventAck** to falling edge of **EventReq**	0		
TKLVH	Delay from falling edge of **EventAck** to next **EventReq**	0		

Full details of each signal are provided in the referenced section.

Analyse Signal used to investigate the state of a T212. The signal brings the processor, links and clocks to a halt within approx three timeslice periods (approx 3 milliseconds). **Reset** may then be applied, which will not destroy state information nor reinitialise the memory interface. After **Reset** has been taken low, **Analyse** may be taken low after which the processor will execute its bootstrap routine. (**2.3**, **3.4**)
Input

BootFromROM If this is held to **VCC**, then the boot program is taken from ROM. Otherwise the T212 awaits a bootstrap message from a link. (**3.3.1**)
Input

CapMinus The negative terminal of an internal power supply used for the internal clocks. This must be connected via a 1 microfarad capacitor to **CapPlus**. If a tantalum capacitor is used **CapMinus** should be connected to the negative terminal. (**8.6**)

CapPlus The positive terminal of an internal power supply used for the internal clocks. This must be connected via a 1 microfarad capacitor to **CapMinus**. If a tantalum capacitor is used **CapPlus** should be connected to the positive terminal.

ClockIn Input clock from which all internal clocks are generated. The nominal input clock frequency for all transputers, of whatever wordlength and speed, is 5 MHz. (**8.6**)
Input

DoNotWire These are output pins reserved for INMOS use. They should be left unconnected.
Output

Error The processor has detected an error and remains high until **Reset**. Errors result from arithmetic overflow, by array bounds violations or by divide by zero. The **Error** flag can also be set by an instruction, to allow other forms of software detected error. (**2.3**)
Output

EventAck The **EventReq** and **EventAck** are a pair of handshake signals for external events. External logic takes **EventReq** high when the logic wishes to communicate with a process in the T212. The rising edge of **EventReq** causes a process to be scheduled. The processor takes **EventAck** high when the process is scheduled. At any time after this point the external logic may take **EventReq** low, following which the processor will set **EventAck** low. (**6**, **8.8**)
Output

EventReq Request external event. See description of **EventAck** above.
Input

GND Power supply return and logic reference, 0 V. There are several **GND** pins to minimize inductance within the package.

HoldToGND These are input pins reserved for INMOS use. They should be held to ground either directly or through a resistor of less than 10K ohms.
Input

HoldToVCC This input pin is reserved for INMOS use. It should be held to the power supply.
Input

LinkIn0 - 3 Input pins of the standard links. A **LinkIn** pin receives a serial stream of bits including protocol bits and data bits. Link inputs must be connected to another Link output or tied to **GND**. The Link input must not be tied high or left floating.
Input

LinkOut0 - 3
Output pins of each of the standard links. A **LinkOut** pin may be left floating or may be connected to one (and only one) **LinkIn** pin. As long as the skew specification is met, the connection may be via buffers. (**4, 8.5**)
Output

LinkSpecial
LinkSpecial selects the non-standard speed to be either 20MBits/sec when high or 5MBits/sec when low. (**4.1.1**)
Input

Link0Special
Link 0 operates at the non-standard speed selected by **LinkSpecial** if this pin is wired high. (**4.1.1**)
Input

Link123Special
Links 1, 2 and 3 operate at the non-standard speed selected by **LinkSpecial** if this pin is wired high. (**4.1.1**)
Input

MemA0 - 15
16-bit wide address bus.
Output

MemD0 - 15
16-bit wide data bus.
Input/Output

MemBAcc
This pin when high causes the memory interface to perform byte access.
Input

MemReq
This input is used by external devices to access the memory interface. When **MemReq** is sampled high, and if there is no processor request outstanding, the address and data lines are taken high impedance and **MemGranted** is taken high.
Input

MemGranted
See description above of **MemReq**.
Output

MemWait
Wait input for the memory interface. If, at the time **MemWait** is sampled, the input is low, the interface cycle proceeds. Otherwise, the interface is held until the input is sampled and found to be low. (**5.4**)
Input

notMemCE
This active low pin is taken low when the address bus is ready for external of the memory cycle.
Output

notMemWrB0-B1
These are the lest significant and most significant bytes respectively. and are active low, write enable pins.
Output

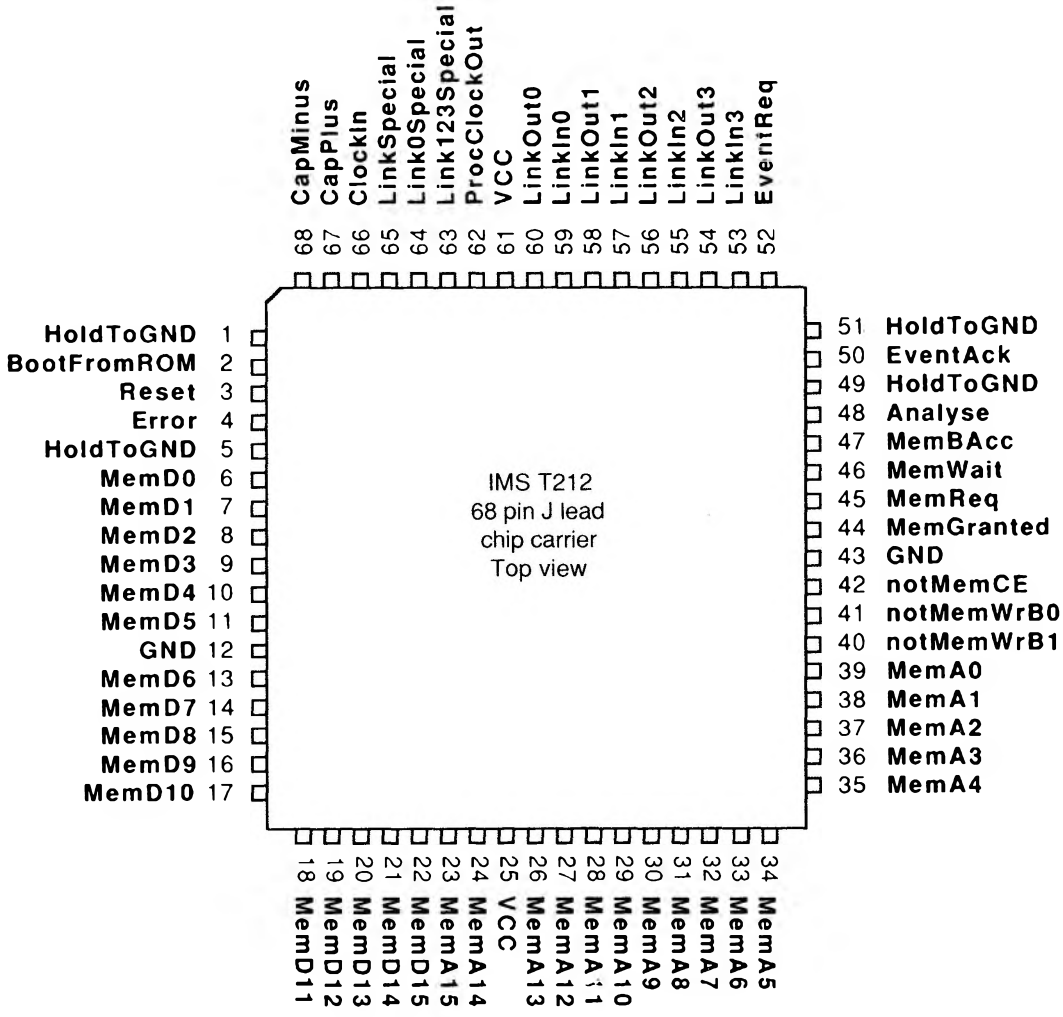
ProcClockOut
The processor clock which is output in phase with the memory interface. The processor clock frequency is a multiple of the input clock frequency. The multiple differs for different speed parts. (**7.1**)
Output

Reset
The falling edge of **Reset** initialises the T212 and then starts the processor executing a bootstrap routine.
Input

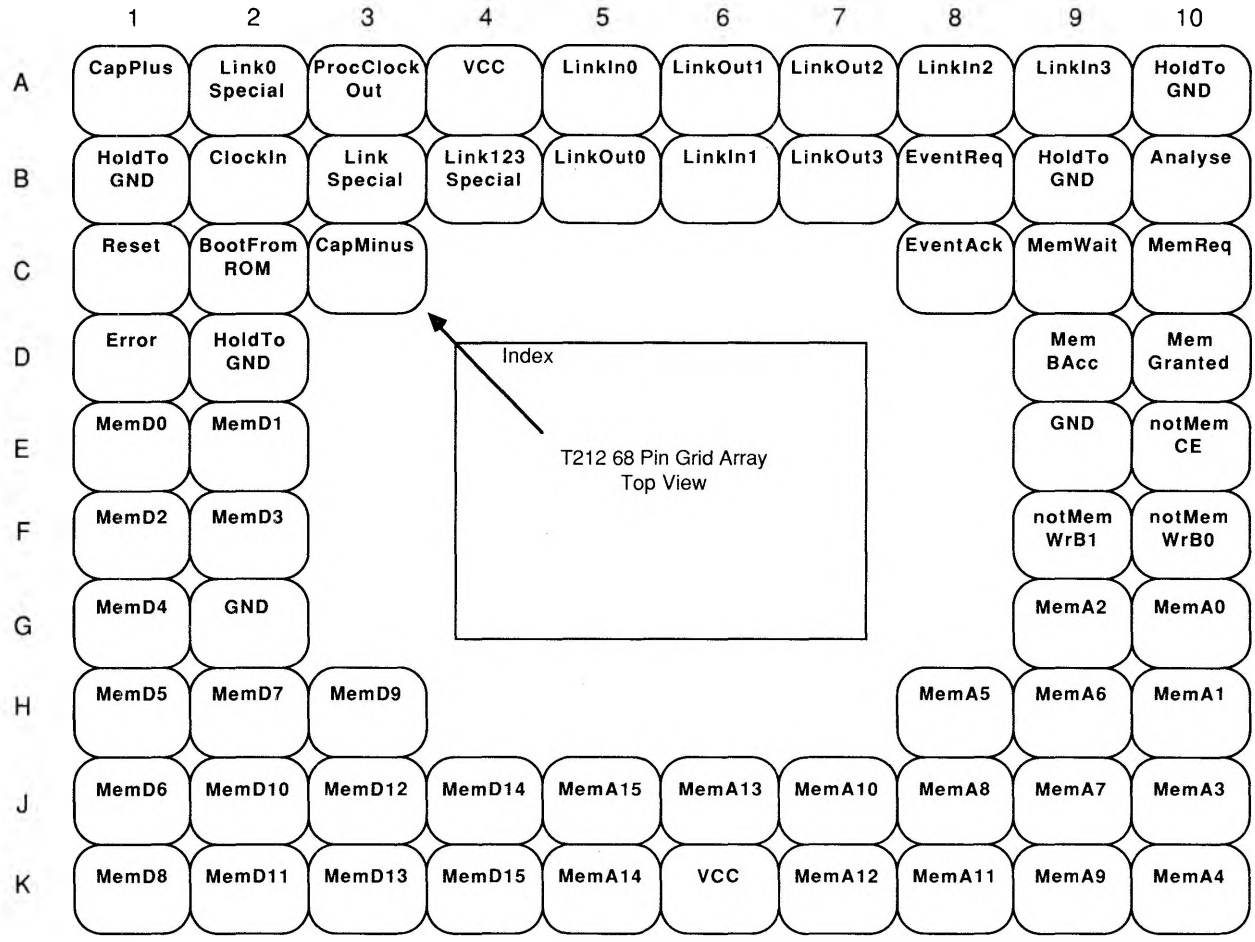
VCC
Power supply, nominally 5 V. There are several **VCC** pins to minimize inductance within the package. **VCC** should be decoupled to **GND** by at least one 100 nF low inductance (such as ceramic) capacitor.

The T212 is available in an 68 pin J-Lead chip carrier or 68 pin Grid Array.

10.1 J-Lead chip carrier



10.2 Pin Grid Array



10.3 Package Dimensions