Engineering data

T-49-17-51

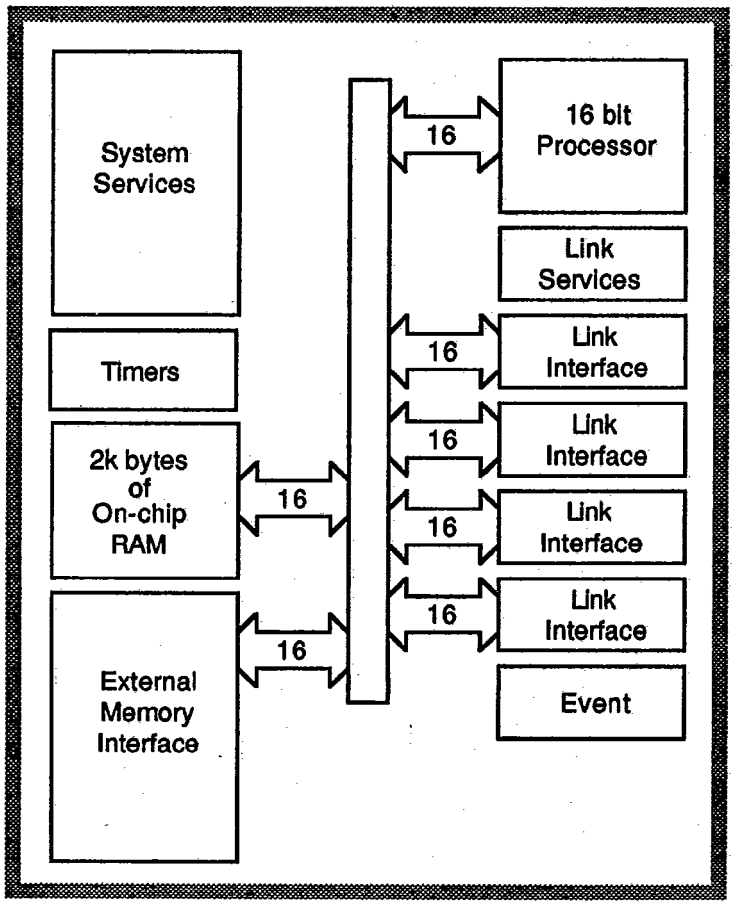
IMS T212M transputer

inmos®

Extended Temperature

FEATURES

Full MIL temperature range (-55°C to +125°C)
MIL-STD-883C processing
16 bit architecture with 8.75 MIPS performance
2 Kbytes 57 ns RAM on chip
Four 5/10/20 Mbits/sec INMOS serial links
16 bit external memory interface
Directly addresses 64 Kbytes at 17.5 Mbytes/sec
Hardware scheduler for concurrent programs
Sub-microsecond context switch
Internal timers for real time processing
External event interrupt
Sub-microsecond typical interrupt latency
Support for run-time error diagnostics
Bootstraps from communication link or ROM
Optional external memory wait states
Internal program continues during DMA
6.4 Mbytes/sec total link data rate
Single 5 MHz clock input
Single +5V ±5% power supply, less than 1 Watt



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

CONTENTS

1 Introduction

The IMS T212M transputer is a MIL-STD-883C compliance 16 bit CMOS microcomputer with 2 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 17.5 MHz achieves an instruction throughput of 8.75 MIPS.

The IMS T212M 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 114 nanoseconds (17.5 Mbytes/sec) for a 17.5 MHz device.

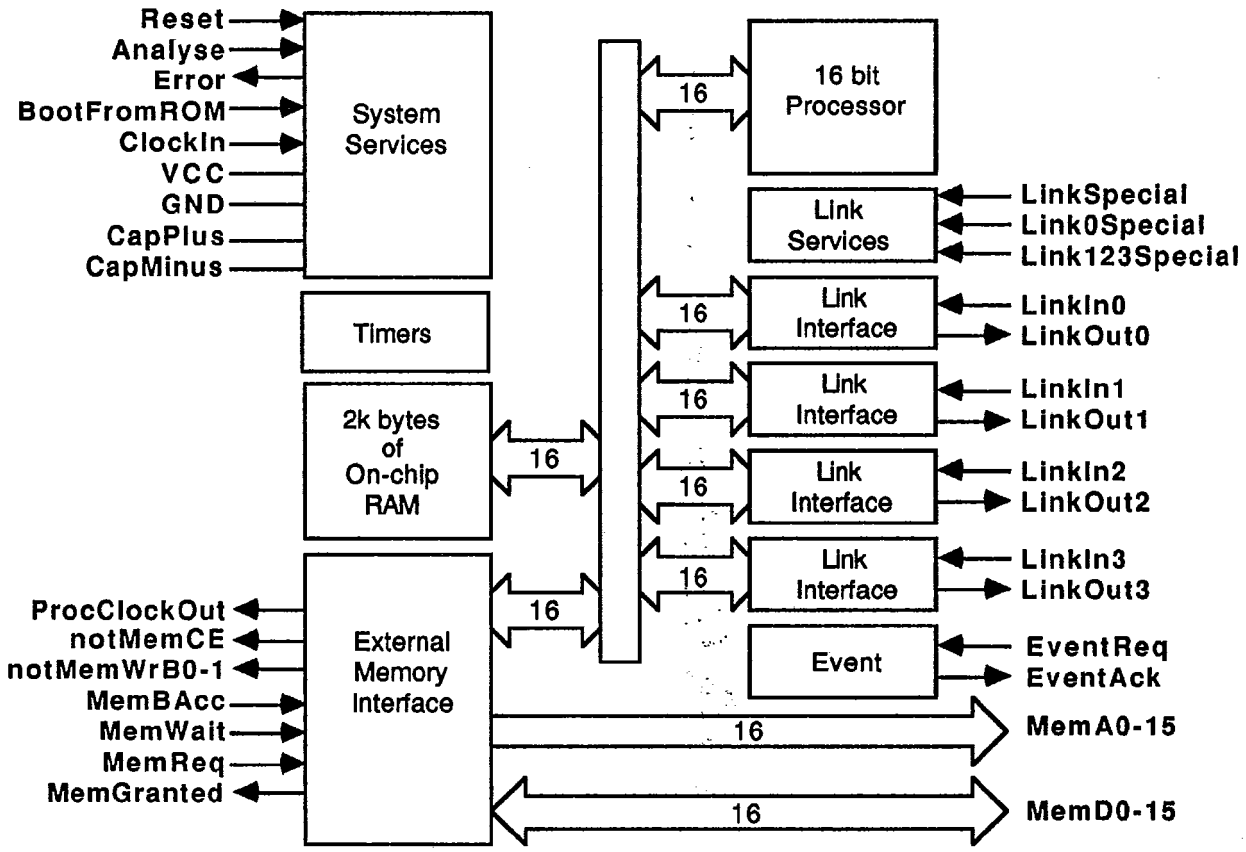
System Services include processor reset and bootstrap control, together with facilities for error analysis.

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 per second, but also operate at 5 or 20 Mbits per second.

The IMS T212M 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 The Transputer Instruction Set - A Compiler Writers' Guide.

This data sheet supplies hardware implementation and characterisation details for the IMS T212M. It is intended to be read in conjunction with the Transputer Reference Manual, which details the architecture of the transputer and gives an overview of occam.

For convenience of description, the IMS T212M operation is split into the basic blocks shown in the Block Diagram.



IMS T212M Block Diagram

2 Pin designations

System Services

Pin	In/Out	Function
VCC, 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	Bootstraps from external ROM or from link
HoldToGND		Must be connected to GND

External Memory Interface

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

Event

Pin	In/Out	Function
EventReq	in	Event request
EventAck	out	Event request acknowledge

Link

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

Notes

Signal names are prefixed by not if they are active low, otherwise they are active high.
Pinout details for various packages are given in section 10.

3 Processor

The 16 bit processor contains instruction processing logic, instruction and work pointers, and an operand register. It directly accesses the high speed 2 Kbyte 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 64 Kbytes of memory via the External Memory Interface (EMI).

3.1 Registers

The design of the transputer processor exploits the availability of fast on-chip memory by having only a small number of registers; six registers 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.

A, **B** and **C** are 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**.

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.

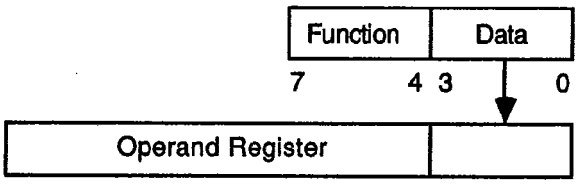
Any location in memory can be accessed relative to the workpointer register, enabling the workspace to be of any size.

Further register details are given in The Transputer Instruction Set - A Compiler Writers' Guide.

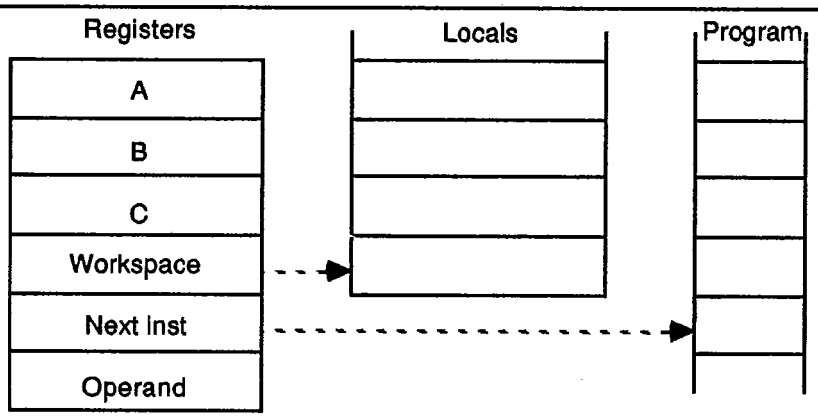
3.2 Instructions

The instruction set has been designed for simple and efficient compilation of high-level languages. All instructions have the same format, designed to give a compact representation of the operations occurring most frequently in programs.

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.



Instruction Format



Registers

3.2.1 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. 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 or Pascal.

3.2.2 Prefix functions

Two more function codes allow the operand of any instruction to be extended in length; *prefix* and *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.

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.

3.2.3 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.

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* and *greater than*. Less frequently occurring operations have encodings which require a single prefix operation.

3.2.4 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

3.2.5 Efficiency of encoding

Measurements show that about 70% of executed instructions are encoded in a single byte (i.e. 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 two instructions for every fetch.

Short instructions also improve the effectiveness of instruction pre-fetch, which in turn improves processor performance. There is an extra word of pre-fetch buffer, so 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.

3.3 Processes and concurrency

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 (section 3.4).

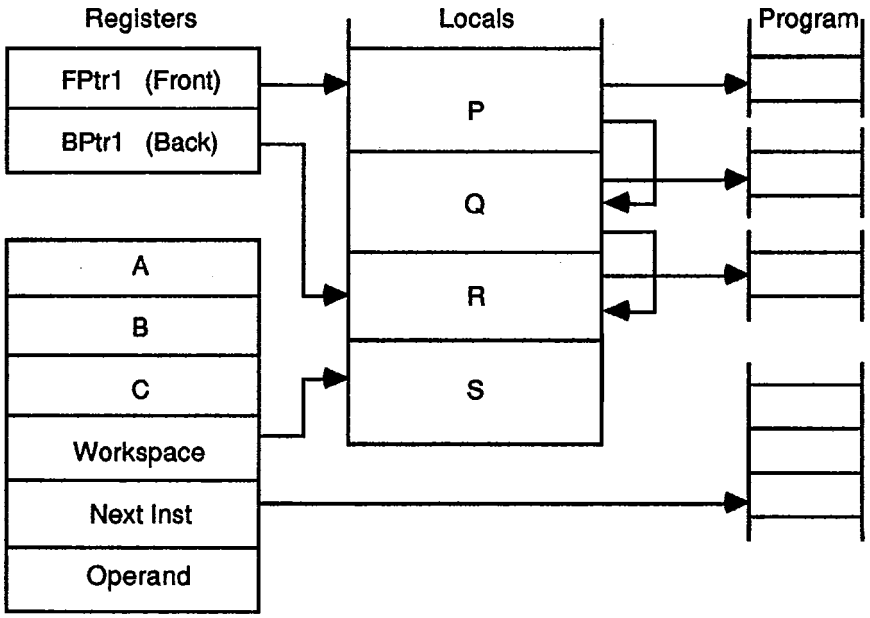
The processor has a microcoded scheduler which enables any number of concurrent processes to be executed together, sharing the processor time. This removes the need for a software kernel.

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. It allocates a portion of the processor's time to each process in turn. Active processes waiting to be executed are held in two linked lists of process workspaces, one of high priority processes and one of low priority processes (section 3.4). Each list is implemented using two registers, one of which points to the first process in the list, the other to the last. In the Linked Process List diagram, process **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.



Linked Process List

High Priority Queue Control Registers

Fptr0	Pointer to front of active process list
Bptr0	Pointer to back of active process list

Low Priority Queue Control Registers

Fptr1	Pointer to front of active process list
Bptr1	Pointer to back of active process list

Each process runs until it has completed its action, but is descheduled whilst waiting for communication from another process or transputer, or for a time delay to complete. 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 3.7.1). The time slice period is 5120 cycles of **ClockIn**, giving ticks approximately 1ms apart.

A process can only be descheduled on certain instructions, known as descheduling points (section 3.7.1). 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, including *start process* and *end process*. When a main process executes a parallel construct, *start process* instructions are used to create the necessary additional 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. When a process is made active it is always added to the end of the list, and thus cannot pre-empt processes already on the same list.

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.

3.4 Priority

The IMS T212M supports two levels of priority. The 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 2n-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 3.7.1).

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

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, a maximum 53 cycles (assuming use of on-chip RAM).

3.5 Communications

Communication between processes is achieved by means of channels. Process communication is point-to-point, synchronised 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* and *output message*.

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

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 *output message* instruction. Data is transferred if the other process is ready. If the channel is not ready or is an external one the process will deschedule.

3.6 Timers

The transputer has two 16 bit timer clocks which 'tick' periodically. The timers provide accurate process timing, allowing processes to deschedule themselves until a specific time.

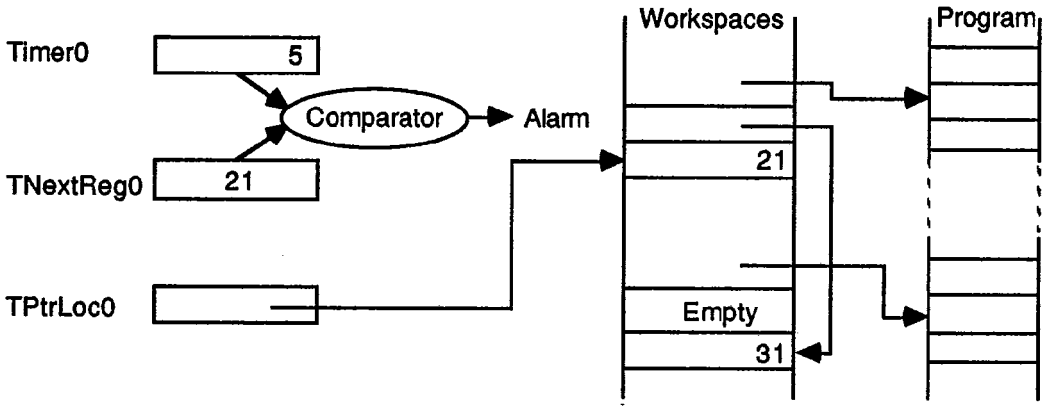
One timer is accessible only to high priority processes and is incremented every microsecond, cycling completely in approximately 65 milliseconds. The other is accessible only to low priority processes and is incremented every 64 microseconds, giving exactly 15625 ticks of this timer in one second. It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

Timer Registers

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

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' then the instruction has no effect. If the time is in the 'future' then the process is descheduled. When the specified time is reached the process is scheduled again.

The Timer Registers diagram shows two processes waiting on the timer queue, one waiting for time 21, the other for time 31.



Timer Registers

3.7 Instruction set summary

The Function Codes table gives the basic function code set (section 3.2.1). Where the operand is less than 16 a single byte encodes the complete instruction. If the operand 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*.

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 #FFE1)		
is coded as		
nfix #1	#6	#61
ldc #1	#4	#41

The Operation Codes tables give details of 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

The Processor Cycles column refers to the number of periods TPCLPCL taken by an instruction executing in internal memory. The number of cycles is given for the basic operation only; where relevant the time for the *prefix* function (one cycle) should be added. For a 17.5 MHz transputer one cycle is 57ns. Some instruction times vary. Where a letter is included in the cycles column it is interpreted from the table below.

b is the bit number of the highest bit set in the operand. Bit 0 is the least significant bit.

n is the number of places shifted.

w is the 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.

The Desch./Error column of the tables indicate if an instruction is a descheduling point (section 3.3) or if it will set **Error** (section 4.8).

3.7.1 Descheduling points

The following instructions are the only ones at which a process may be descheduled (section 3.3). They are also the ones at which the processor will halt if **Analyse** is asserted (section 4.7).

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

3.7.2 Error instructions

The following instructions are the only ones which can affect **Error** (section 4.8).

add	*add constant*
subtract	*multiply*
divide	*remainder*
long add	*long subtract*
long divide	
set error	*testerr*
check word	*check subscript from 0*
check single	*check count from 1*

Function Codes

Function Code	Memory Code	Mnemonic	Processor Cycles	Name	Desch/ Error
0	0X	j	3	jump	Desch
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	Error
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	

Arithmetic/Logical Operation Codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	Desch/ Error
46	24F6	and	1	and	
4B	24FB	or	1	or	
33	23F3	xor	1	exclusive or	
32	23F2	not	1	bitwise not	
41	24F1	shl	n+2	shift left	
40	24F0	shr	n+2	shift right	
05	F5	add	1	add	Error
0C	FC	sub	1	subtract	Error
53	25F3	mul	26	multiply	Error
2C	22FC	div	23	divide	Error
1F	21FF	rem	21	remainder	Error
09	F9	gt	2	greater than	
04	F4	diff	1	difference	
52	25F2	sum	1	sum	
08	F8	prod	b+4	product	

Long Arithmetic Operation Codes

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

General Operation Codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	Desch/ Error
00	F0	rev	1	reverse	
3A	23FA	xword	4	extend to word	
56	25F6	cword	5	check word	Error
1D	21FD	xdble	2	extend to double	
4C	24FC	csngl	3	check single	Error
42	24F2	mint	1	minimum integer	

Indexing/Array Operation Codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	Desch/ Error
02	F2	bsub	1	byte subscript	
0A	FA	wsub	2	word subscript	
34	23F4	bcnt	2	byte count	
3F	23FF	wcnt	4	word count	
01	F1	lb	5	load byte	
3B	23FB	sb	5	store byte	
4A	24FA	move	2w+8	move message	

Timer Handling Operation Codes

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

Input/Output Operation Codes

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

Control Operation Codes

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

Scheduling Operation Codes

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

Error Handling Operation Codes

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

Processor Initialisation Operation Codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	Desch/ Error
2A	22FA	testpranal	2	test processor analysing	
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	

4 System Services

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

4.1 Power

Power is supplied to the transputer via the **VCC** 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 100nF low inductance (e.g. ceramic) capacitor between **VCC** 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 **VCC** 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

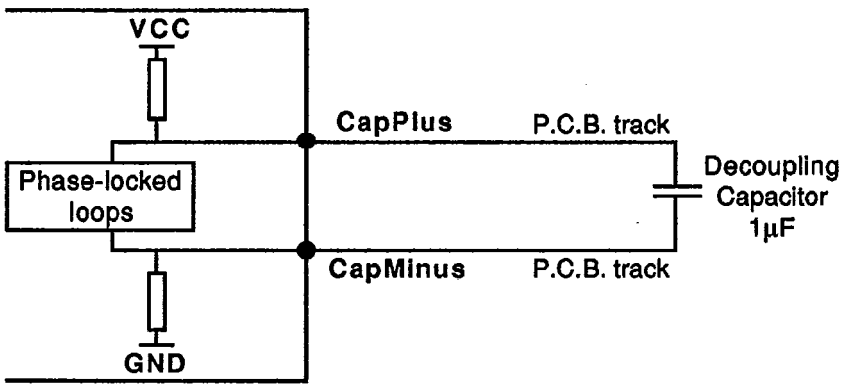
The internally derived power supply for internal clocks requires an external low leakage, low inductance 1μF capacitor to be connected between **CapPlus** and **CapMinus**. A ceramic capacitor is preferred, with an impedance less than 3 ohms between 100 KHz and 20 MHz. If a polarised capacitor is used the negative terminal should be connected to **CapMinus**. Total PCB track length should be less than 50mm. The connections must not touch power supplies or other noise sources.

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 5MHz, regardless of 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.



Recommended PLL Decoupling

Input Clock

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TDCLDCH	ClockIn pulse width low	40			ns	
TDCHDCL	ClockIn pulse width high	40			ns	
TDCLDCL	ClockIn period		200		ns	1,3
TDCerror	ClockIn timing error			±0.5	ns	2
TDC1DC2	Difference in ClockIn for 2 linked devices			400	ppm	3
TDCr	ClockIn rise time			10	ns	4
TDCf	ClockIn fall time			8	ns	4

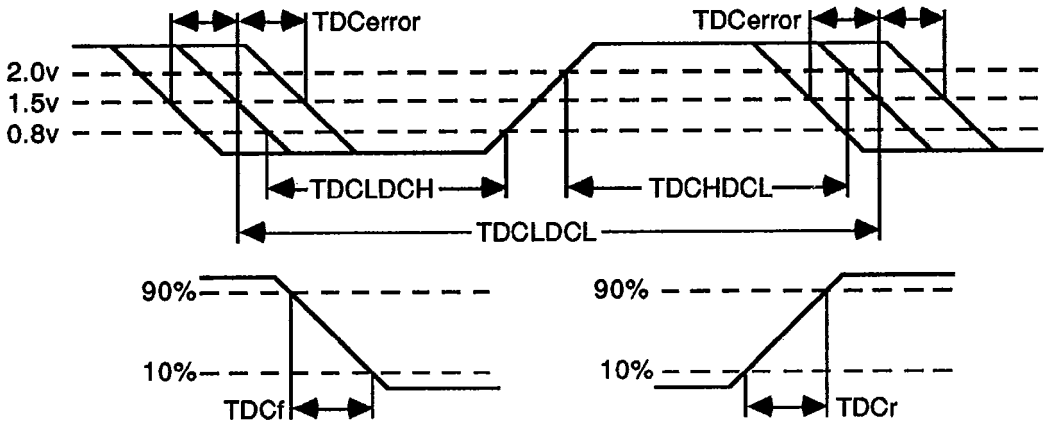
Notes

1 Measured between corresponding points on consecutive falling edges.

2 Variation of individual falling edges from their nominal times.

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

4 Clock transitions must be monotonic within the range VIH to VIL



ClockIn Timing

4.4 Reset

Reset can go high with **VCC**, but must at no time exceed the maximum specified voltage for **VIH**. After **VCC** 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 (section 6.10).

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.

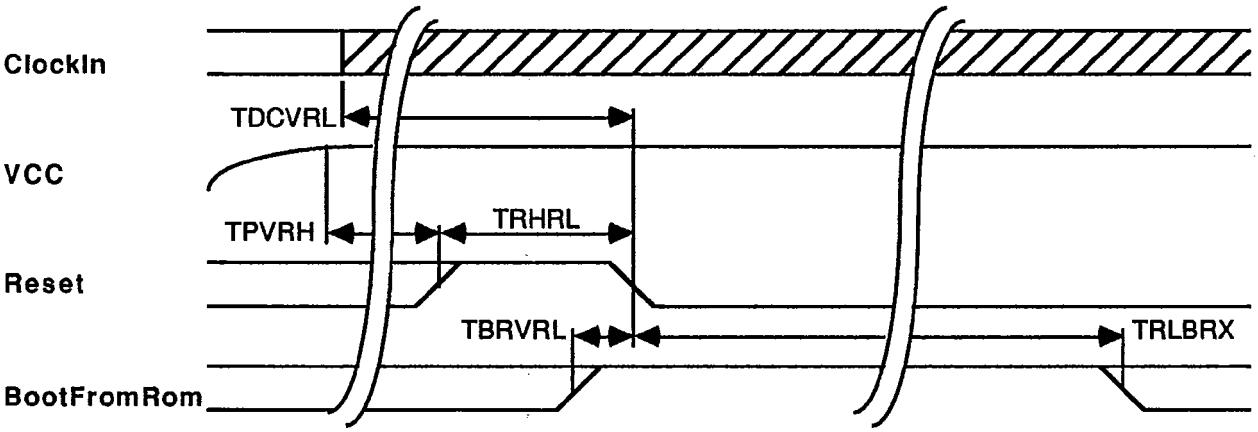
Reset, Analyse

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
TAHRH	Analyse setup before Reset	3			ms	
TRLAL	Analyse hold after Reset end	1			ns	
TBRVRL	BootFromRom setup	0			ms	
TRLBRX	BootFromRom hold after Reset	50			ms	
TALBRX	BootFromRom hold after Analyse	50			ms	

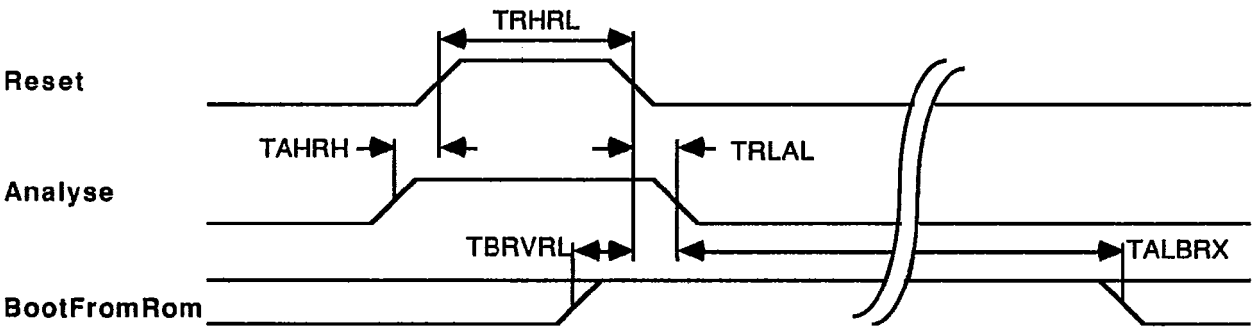
Notes

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

2 At power-on reset.



Reset Timing with Analyse Low



Reset and Analyse Timing

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.

If **BootFromRom** is connected high (e.g. to **VCC**) 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. The processor is in the low priority state. The **W** register points to **MemStart** (section 5).

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. The memory space immediately above the loaded code is used as work space. Messages arriving on other links after the control byte has been received and on the bootstrapping link after the last bootstrap byte will be retained 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 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.

4.7 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (section 3.7.1). 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.

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 following values:

I **MemStart** if bootstrapping from a link, or the external memory bootstrap address (#7FFE) 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.

4.8 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 (section 3.7.2). Once set, the **Error** flag is only cleared by executing the instruction *testerr*. It is not cleared by **Reset** in order that analysis can identify any errant transputer.

A process can be programmed to stop if **Error** 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 **Reset** and **Analyse** to function with the flags in indeterminate states.

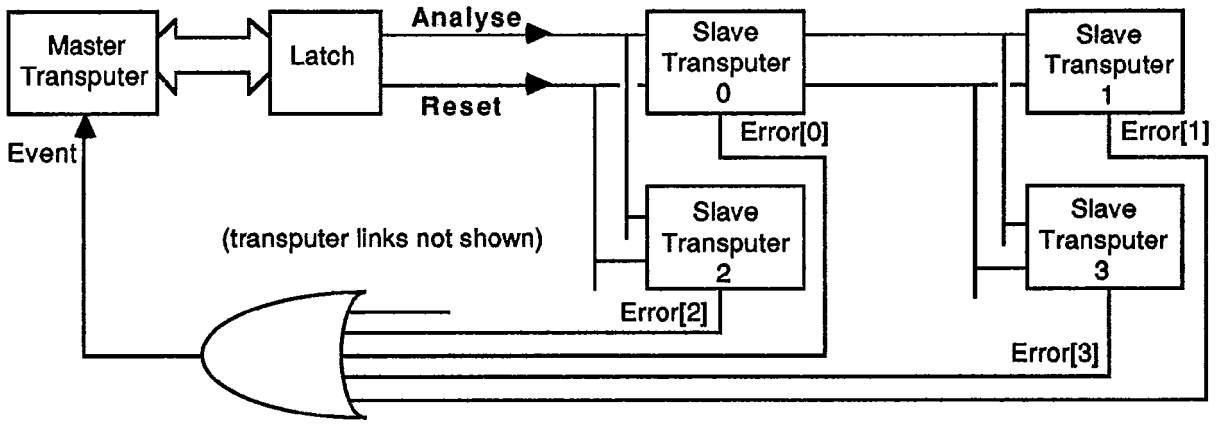
An alternative method of error handling is to have the errant process or transputer cause all transputers to halt. This can be done by applying the **Error** output signal of the errant transputer to the **EventReq** pin of a suitably programmed master transputer. Since the process state is preserved when stopped by an error, the master transputer can then use the **Analyse** function to debug the fault.

Error checks can be removed completely to optimise the performance of a proven program; any unexpected error then occurring will have an 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 **Error** changing from 0 to 1 whilst **HaltOnError** is set, register I points two bytes past the instruction which set **Error**. After halting due to **Analyse** being taken high, register I points one byte past the instruction being executed. In both cases I will be copied to register **A**.



Error Handling in a Multi-Transputer System

5 Memory

The IMS T212M has 2 Kbytes of fast internal memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (section 6.1). The transputer can also access an additional 62 Kbytes of external memory space. Internal and external memory are part of the same linear address space.

IMS T212M 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.

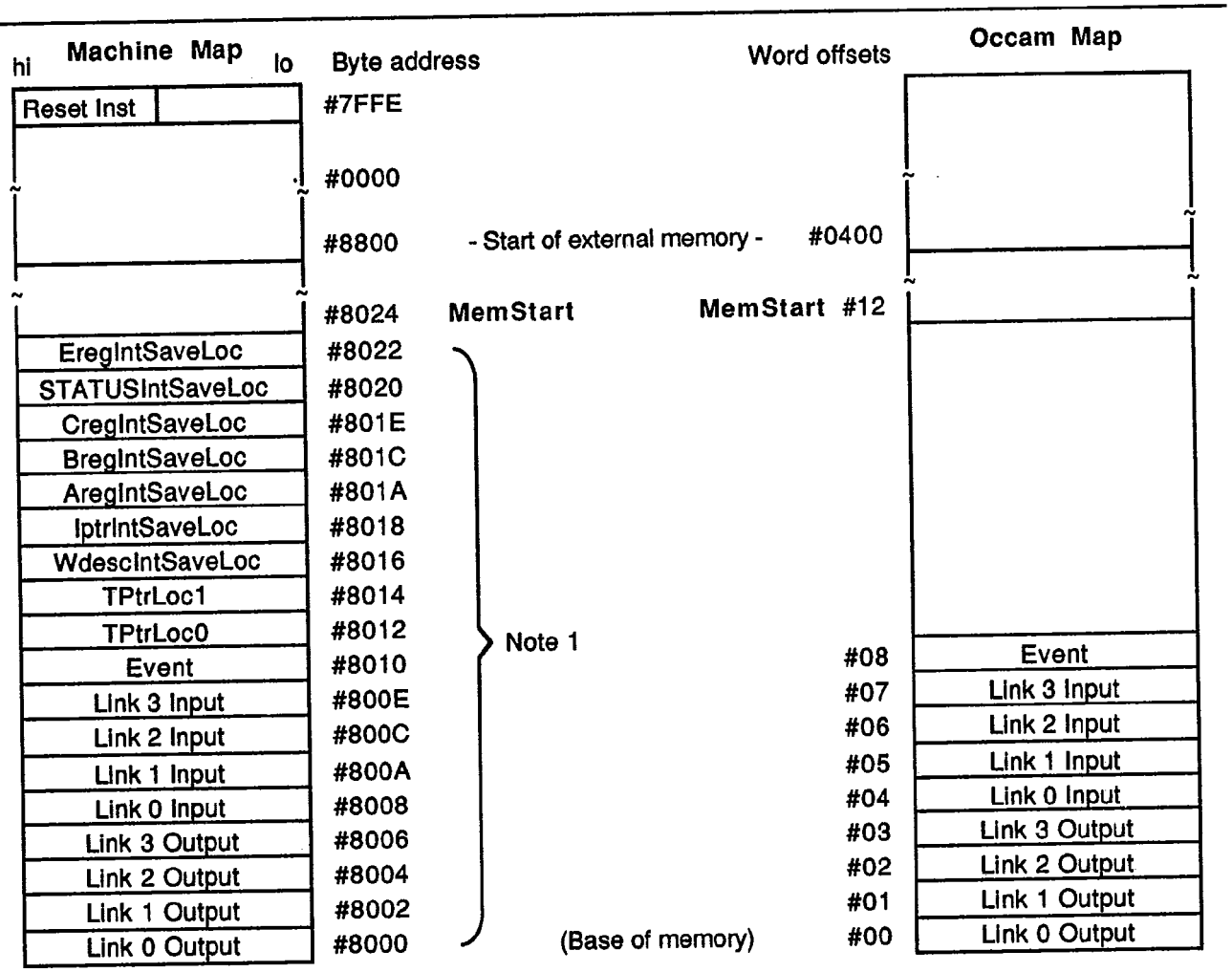
Internal memory starts at the most negative address #8000 and extends to #87FF. User memory begins at #8024; this location is given the name **MemStart**.

A reserved area at the bottom of internal memory 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 #8800 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.



Memory Map

Notes

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 (see **Analyse** section 4.7). For details see The Transputer Instruction Set - A Compiler Writers' Guide.

6 External Memory Interface

The 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 (section 6.8).

6.1 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 (section 12).

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

T-49-17-51

6.2 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, displaced by approximately one fourth of a cycle from **ProcClockOut** edges. **T2** can be extended indefinitely by adding externally generated wait states of one complete processor cycle each.

An external memory cycle is always a complete number of cycles **TPCLPCL** in length. The start of **T1** always coincides with the low phase of **ProcClockOut**.

ProcClockOut

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TPCLPCL	ProcClockOut period	a-1	a	a+1	ns	1
TPCHPCL	ProcClockOut pulse width high	b-2.5	b	b+2.5	ns	2
TPCLPCH	ProcClockOut pulse width low		c		ns	3
TPCstab	ProcClockOut stability			4	%	4

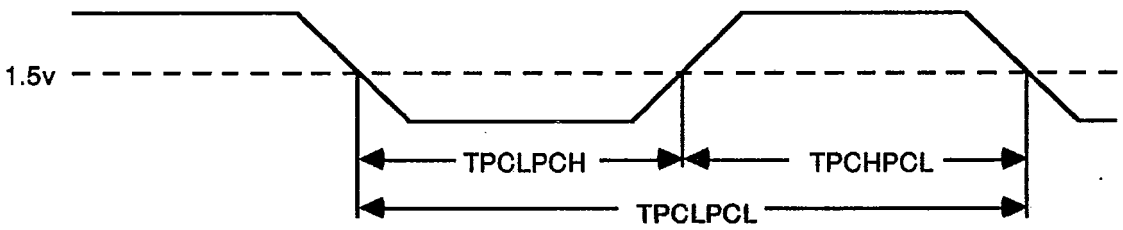
Notes

1 a is **TDCLDCL/PLLx**.

2 b is 0.5∗**TPCLPCL** (half the processor clock period).

3 c is **TPCLPCL-TPCHPCL**.

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



ProcClockOut Timing

6.3 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** reflect the internal read/write operation, **notMemCE** is 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 conjuction with peek and poke (section 4.6).

6.4 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 T212M 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.

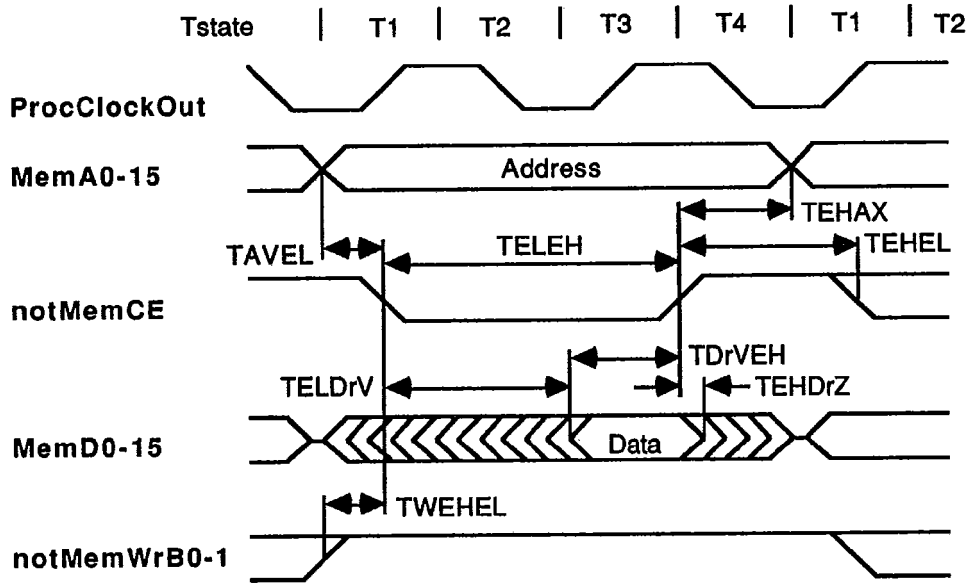
Read

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TAVEL	Address valid before chip enable low	15		19	ns	
TELEH	Chip enable low	65		72	ns	
TEHEL	Delay before chip enable re-assertion	40		51	ns	1
TEHAX	Address hold after chip enable high	21		27	ns	
TELDrV	Data valid from chip enable low			43	ns	
TDrVEH	Data setup before chip enable high	15			ns	
TEHDrZ	Data hold after chip enable high	0			ns	
TWEHEL	Write enable setup before chip enable low	18			ns	2

Notes

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

2 Timing is for both write enables **notMemWrB0-1**.



External Read Cycle

6.5 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 T212M reads it during **T3**. Data can be removed any time during **T4**, but must be off the bus no later than the end of that period.

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 (section 6.8).

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** (section 6.9).

6.6 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 active before the chip enable signal **notMemCE** becomes active, thus reducing memory access time and the risk of bus contention.

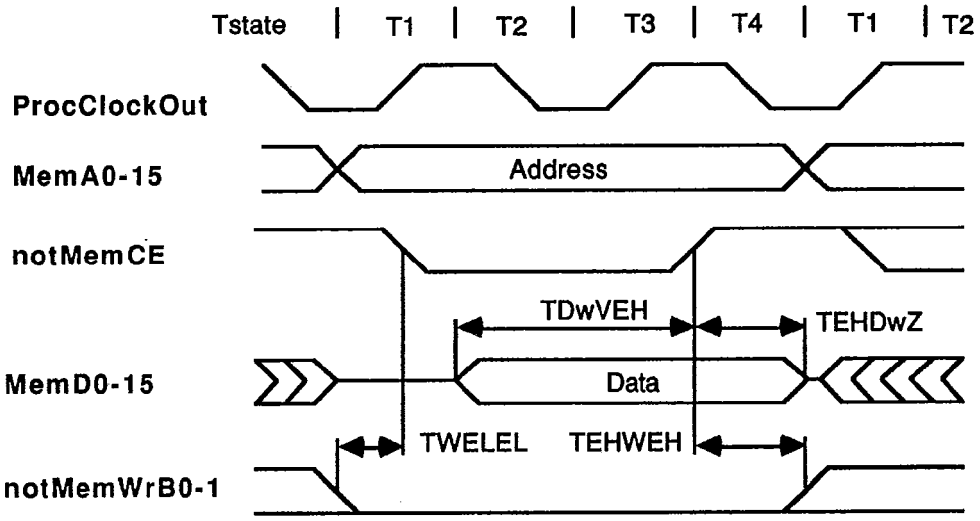
Data must be strobed into memory by, or in conjunction with, **notMemCE**, as the write enables are not guaranteed to go high between consecutive write cycles. The write enables are placed in a high impedance state during DMA.

Write

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TDwVEH	Data setup before chip enable high	42			ns	
TEHDwZ	Data hold after write	24		32	ns	
TWELEL	Write enable setup before chip enable low	4		24	ns	1
TEHWEH	Write enable hold after chip enable high	18		27	ns	1

Notes

1 Timing is for both write enables **notMemWrB0-1**.



External Write Cycle

6.7 notMemCE

The active low signal **notMemCE** is used to enable external memory on both read and write cycles. It must be used, in conjunction with the write enables **notMemWrB0-1**, to write data into memory; the write enable lines only select the byte of memory to be written.

6.8 MemBAcc

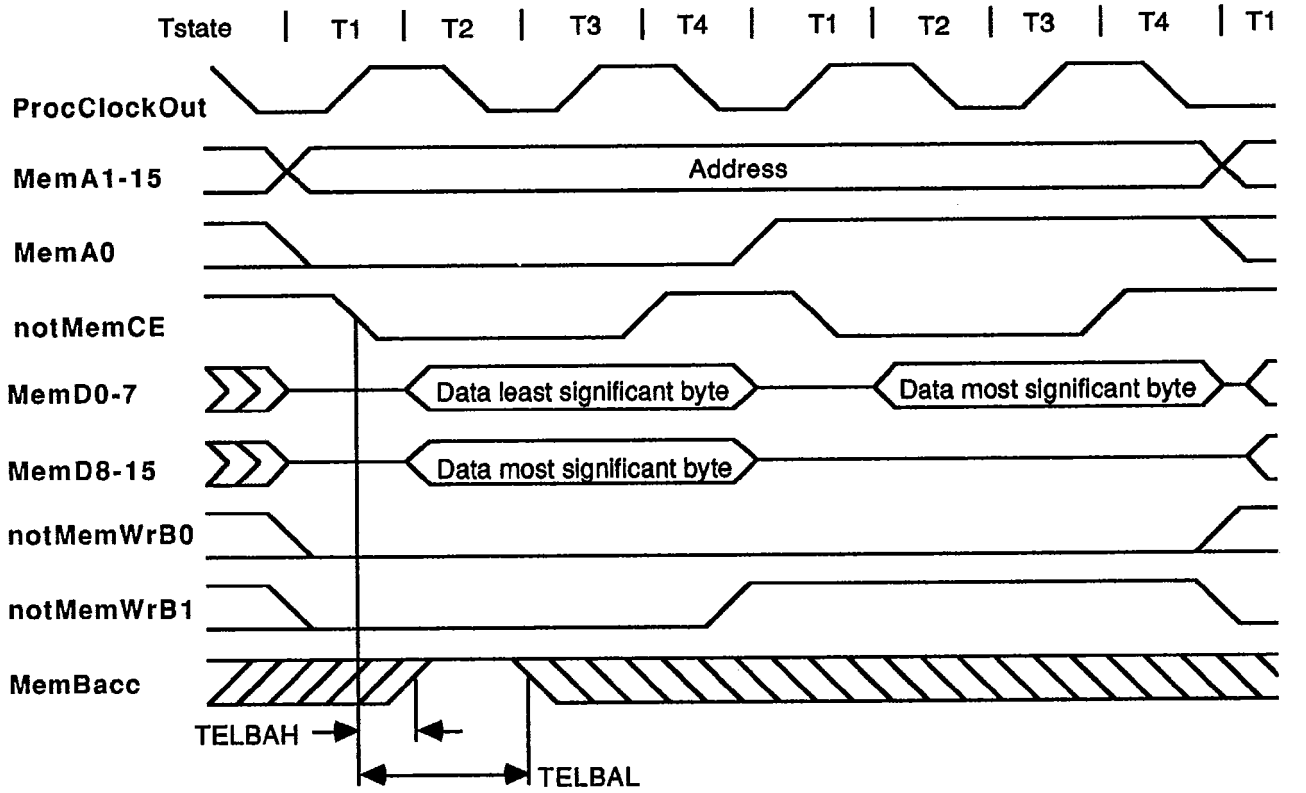
The IMS T212M will, by default, perform word access at even memory locations. Access to byte-wide memory can be achieved by taking **MemBAcc** high with the timing shown. Where all external memory operations are to byte-wide memory, **MemBAcc** may be wired permanently high. The state of this signal is latched during **T2**.

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).

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 **notMemWrB1** remains high, both for read and write, and **MemD8-15** are high impedance. When writing a single byte with **MemBAcc** high, both the first and second cycles are performed with **notMemWrB0** asserted in the appropriate cycle.

Byte-Wide Memory Access

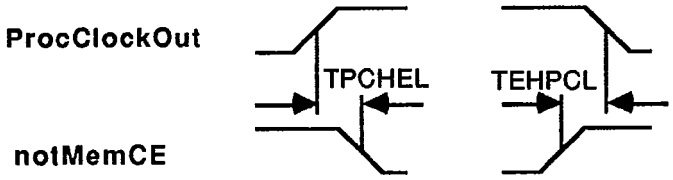
SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TELBAH	MemBAcc high from chip enable			15	ns	
TELBAL	MemBAcc low from chip enable	29			ns	



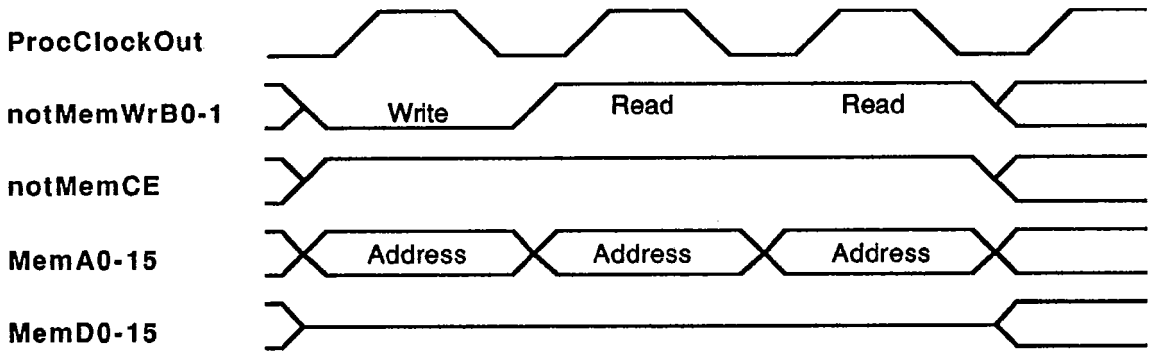
Word Write to Byte-Wide Memory

notMemCE to ProcClockOut Skew

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TPCHEL	Chip enable falling from ProcClockOut rising	2		8	ns	
TEHPCL	Chip enable rising to ProcClockOut falling	10		15	ns	



Skew of notMemCE to ProcClockOut



Typical Bus Activity for Internal Memory Cycles

6.9 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 near the falling edge of **ProcClockOut** during **T2**, and should not change state in this region. If **MemWait** is still high when sampled near the falling edge of **ProcClockOut** in **W2** then another wait period will be inserted. This can continue indefinitely.

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

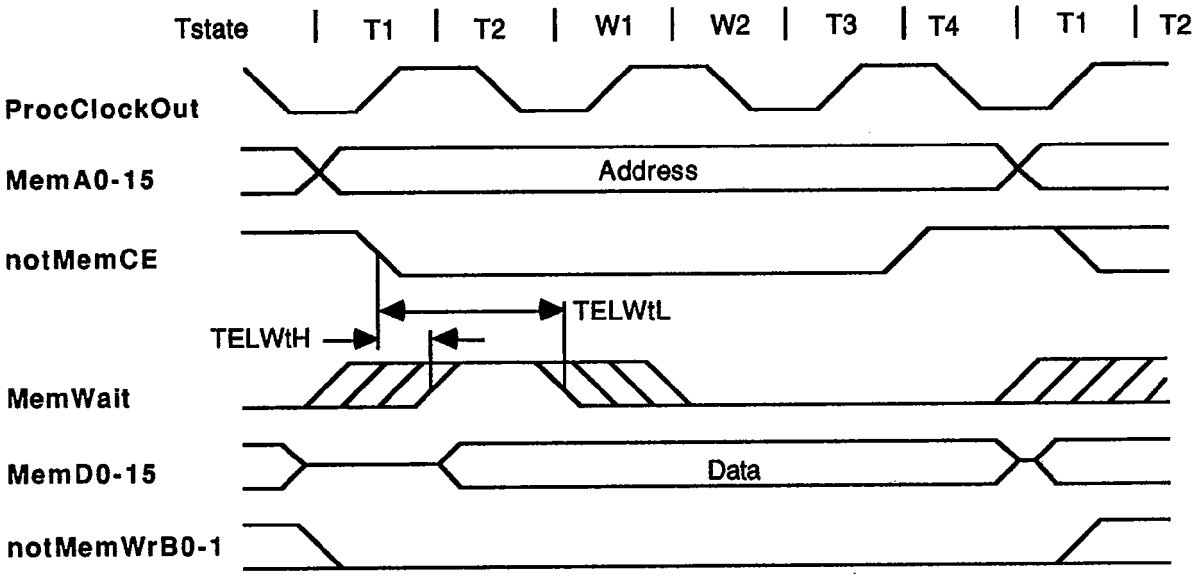
The **Programmable Wait State Generator** circuit shown is designed to be interfaced directly to any memory or peripheral enable signal; 'F' series devices should be employed to ensure minimum delay between **notMemCE** and a valid **notWaitX** input. Only one wait select input line should be low at any one time; for zero wait states **notWait0** must be asserted.

Memory Wait

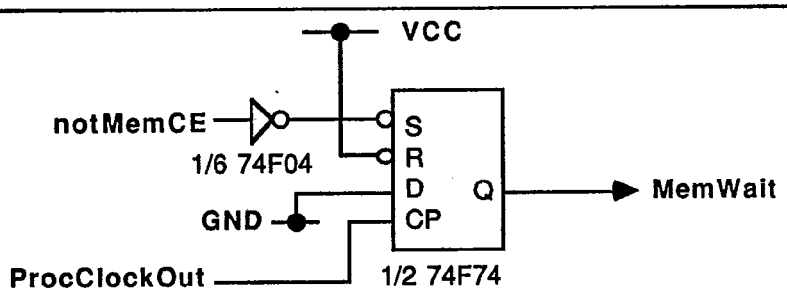
SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TELWtH	MemWait asserted after chip enable low			13	ns	
TELWtL	Wait hold after chip enable low	23		a*b+13	ns	1

Notes

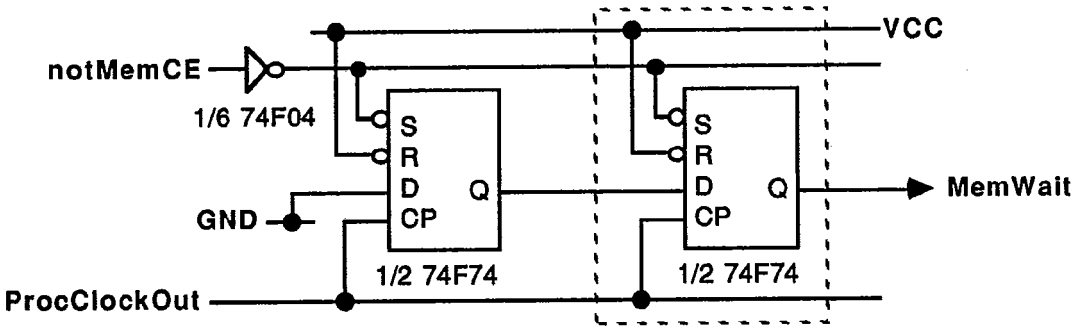
1 a is number of wait states. b is toleranced clock period; 56 ns for IMS T212M-17.



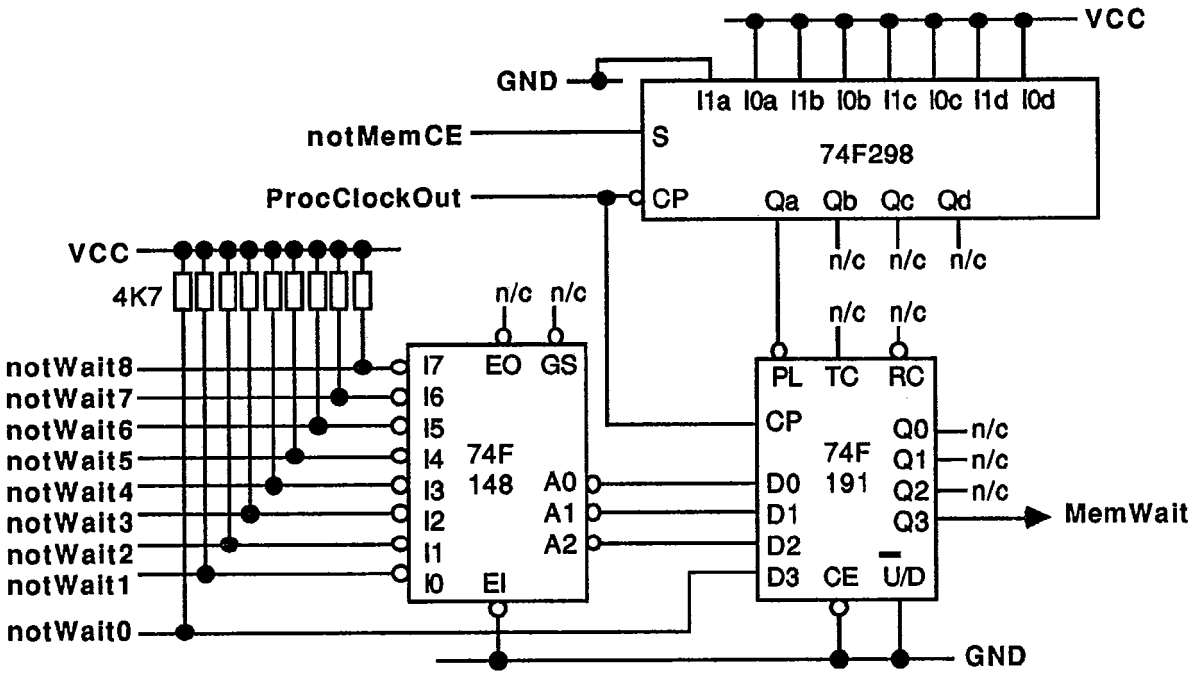
Memory Wait Timing



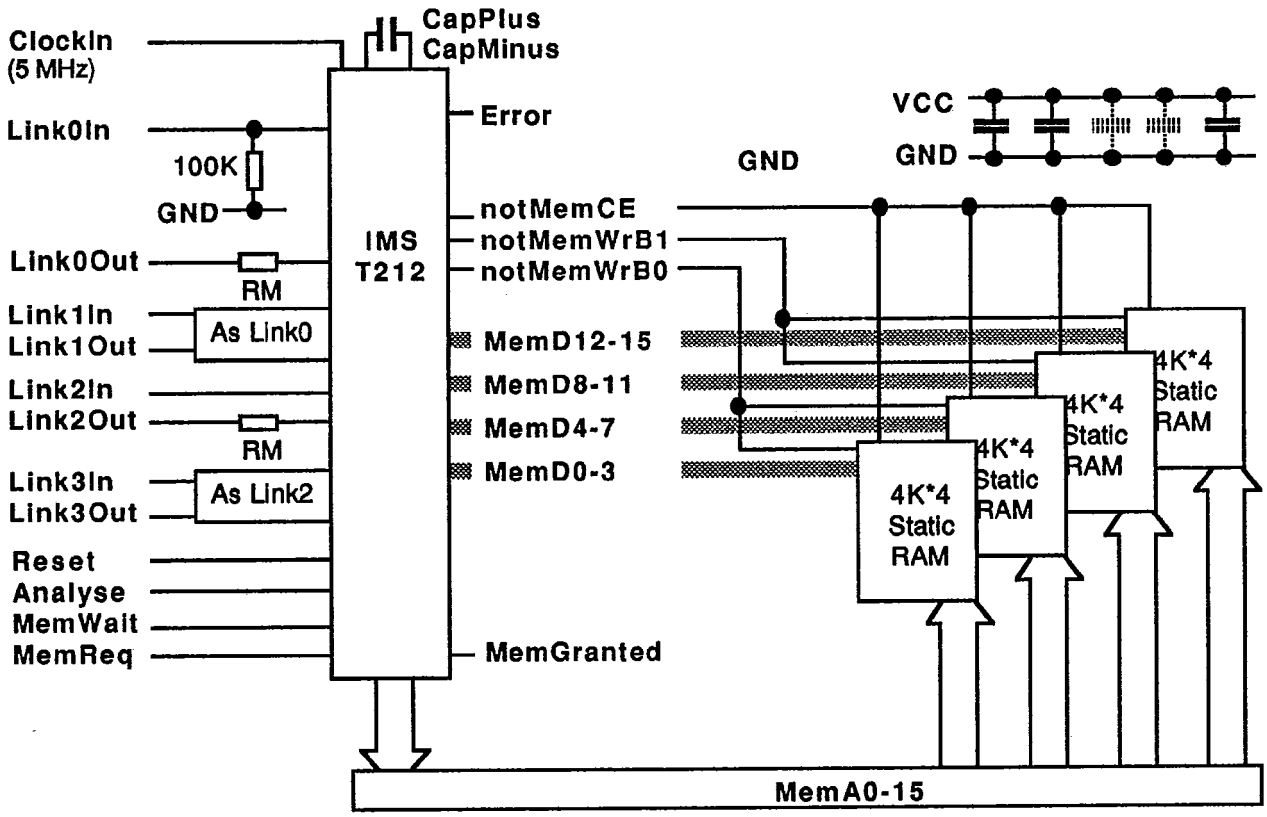
Single Wait State Generator



Extendable Wait State Generator



Programmable Wait State Generator



IMS T212M Application

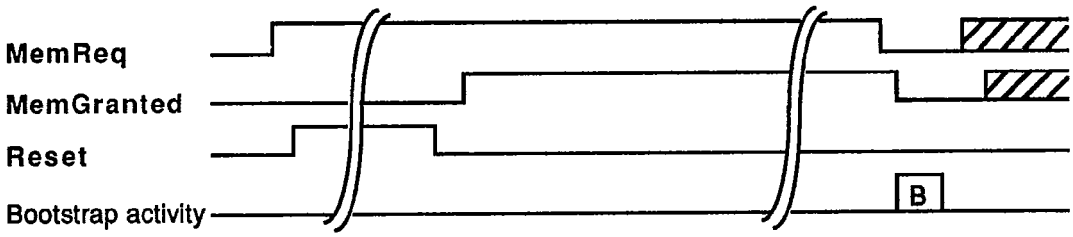
6.10 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. For external memory cycles, the IMS T212M 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 high phase of **ProcClockOut**. **MemA0-15, MemD0-15, notMemWrB0-1** and **notMemCE** are tristated before **MemGranted** is asserted.

Removal of **MemReq** is sampled during each high phase 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 VCC 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.

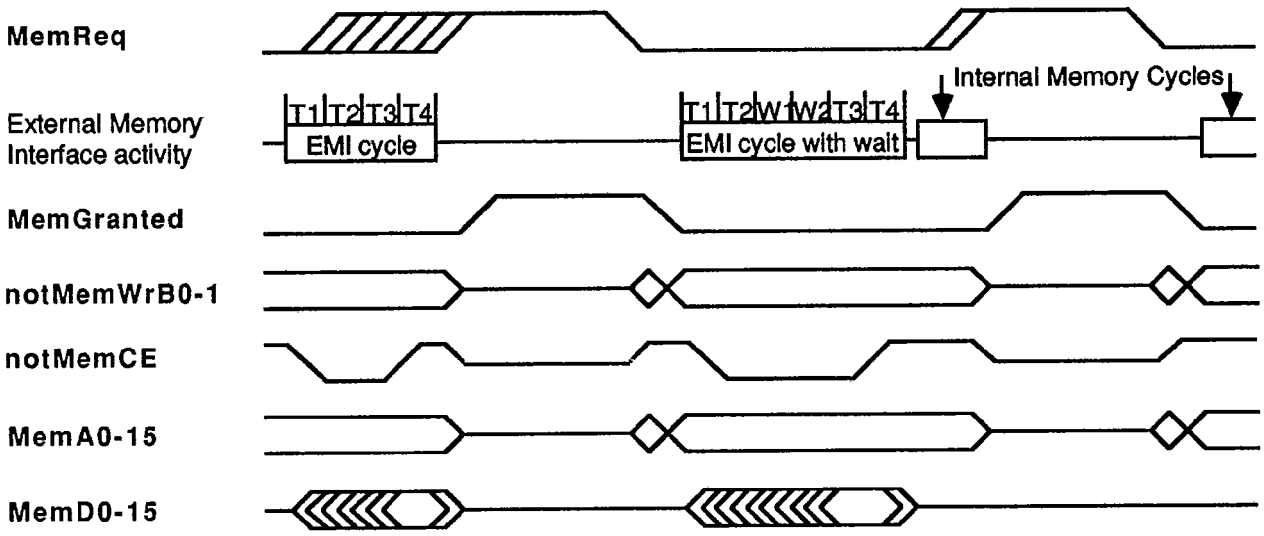
DMA allows a bootstrap program to be loaded into external RAM ready for execution after reset. If **MemReq** is held high throughout reset, **MemGranted** will be asserted before the bootstrap sequence begins. **MemReq** must be high at least one period **TDCLDCL** of **ClockIn** before **Reset**. The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



DMA sequence at Reset

Notes

B Bootstrap sequence.



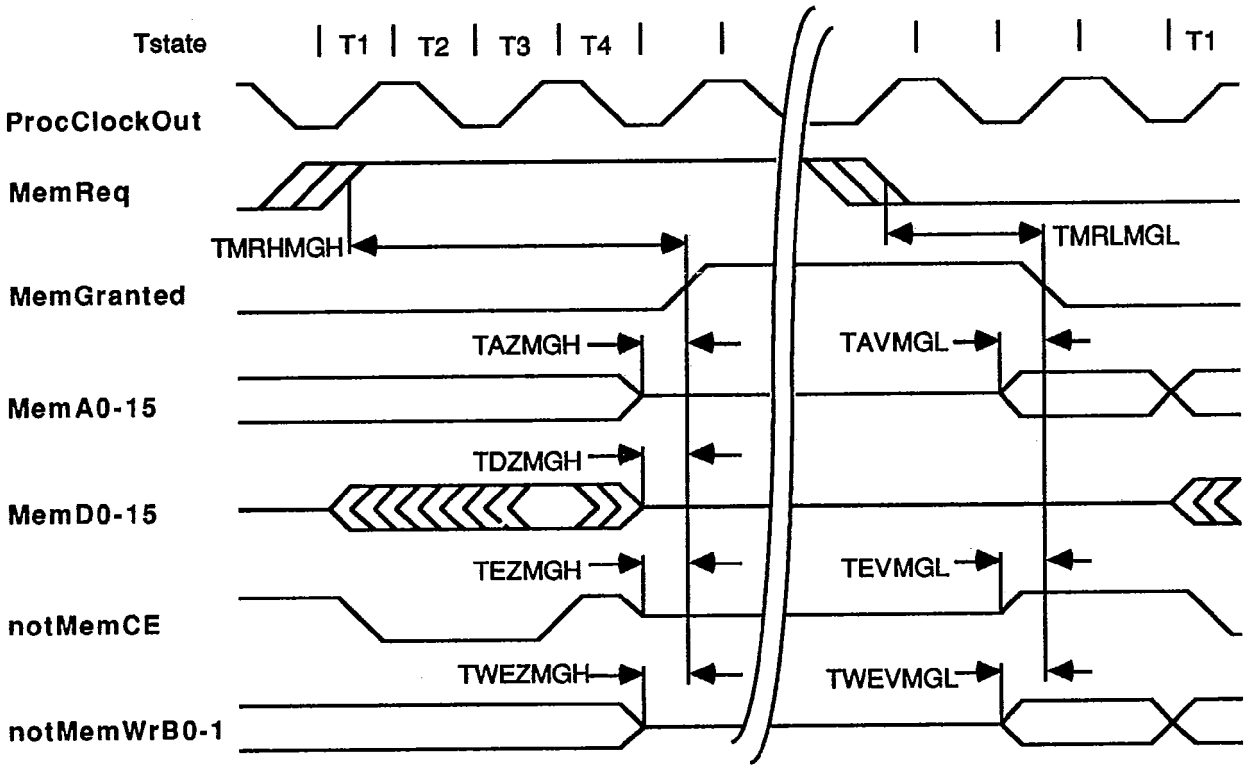
Operation of MemReq and MemGranted with External and Internal Memory Cycles

Memory Request

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TMRHMGH	Memory request response time	100		a	ns	1
TMRLMGL	Memory request end response time	100		114	ns	
TAZMGH	Address bus tristate before memory granted	0			ns	
TAVMGL	Address bus active after memory granted end	0			ns	
TDZMGH	Data bus tristate before memory granted	0			ns	
TEZMGH	Chip enable tristate before memory granted	0			ns	
TEVMGL	Chip enable active after memory granted end	0			ns	
TWEZMGH	Write enable tristate before memory granted	0			ns	
TWEVMGL	Write enable active after memory granted end	0			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.



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.

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.

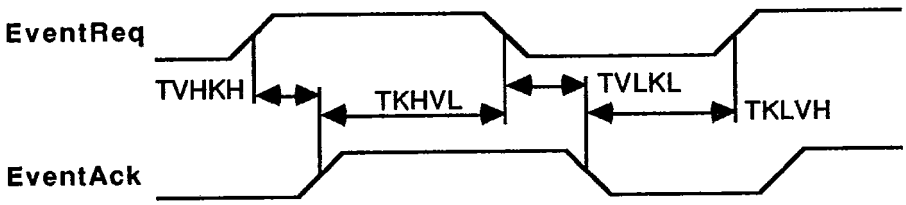
If the process is a high priority one and no other high priority process is running, the maximum latency is 53 processor cycles **TPCLPCL**, assuming all memory accesses are internal ones. Typical latency is 19 processor cycles. Setting a high priority task to wait for an event input is a way of interrupting a transputer program.

Event

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TVHKH	Event request response	0			ns	
TKHVL	Event request hold	0			ns	
TVLKL	Delay before removal of event acknowledge	0		a	ns	1
TKLVH	Delay before re-assertion of event request	0			ns	

Notes

1 a is **TPCLPCL** .



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 T212M links support the standard INMOS communication speed of 10 Mbits per second. In addition they can be used at 5 or 20 Mbits per second. 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

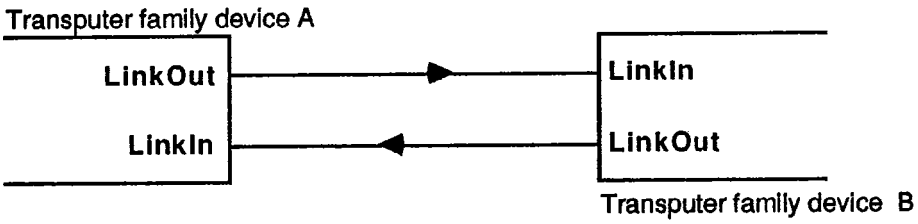
via a backplane. Direct connection may be made between devices separated by a distance of less than 300 millimetres. For longer distances a matched 100 Ohm transmission line should be used with series matching resistors **RM**. When this is done the line delay should be less than 0.4 bit time to ensure 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.

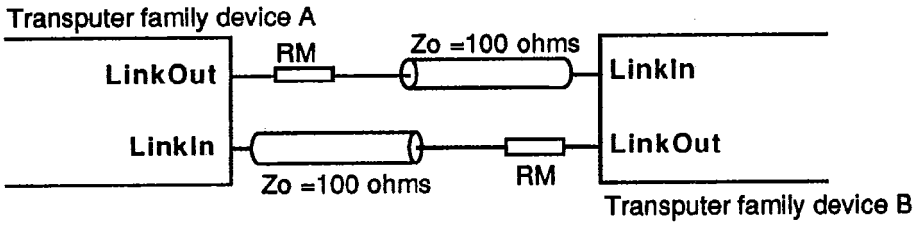
Link speeds can be set by **LinkSpecial**, **Link0Special** and **Link123Special**. The link 0 speed can be set independently. The table shows uni-directional and bi-directional data rates in Kbytes/second for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed 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.

Speed settings for Links

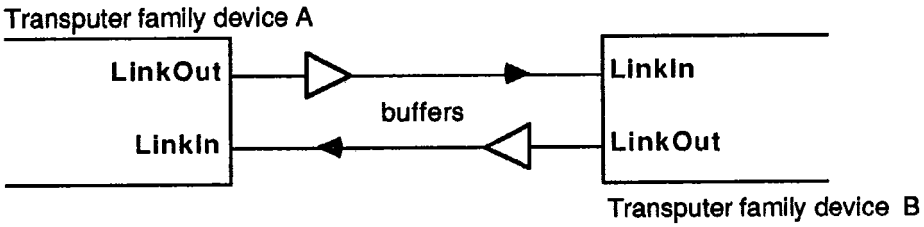
Link Special	Linkn Special	Mbits /sec	Kbytes/sec Uni	Kbytes/sec Bi
0	0	10	400	800
0	1	5	200	400
1	0	10	400	800
1	1	20	800	1600



Links Directly Connected



Links Connected by Transmission Line



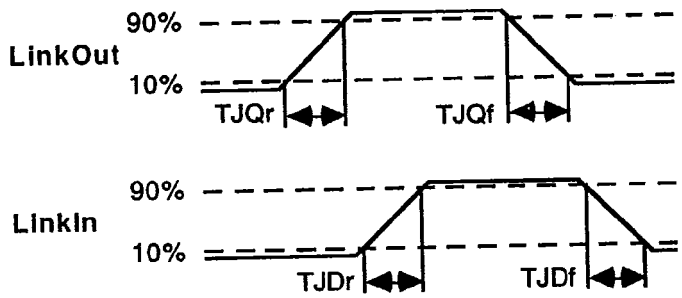
Links Connected by Buffers

Link

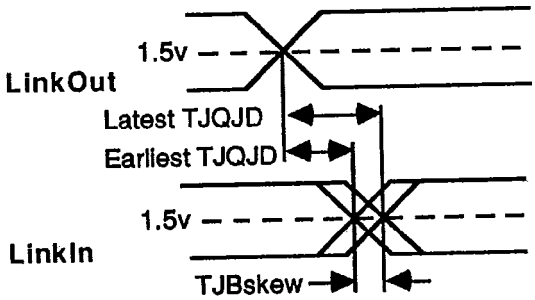
SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TJQr	LinkOut rise time			20	ns	
TJQf	LinkOut fall time			10	ns	
TJDr	LinkIn rise time			20	ns	
TJDf	LinkIn fall time			20	ns	
TJQJD	Buffered edge delay	0			ns	
TJBskew	Variation in TJQJD 5 Mbits/s			30	ns	1
	10 Mbits/s			10	ns	1
	20 Mbits/s			3	ns	1
CLIZ	LinkIn capacitance @ f=1MHz			7	pF	
CLL	LinkOut load capacitance			50	pF	
RM	Series resistor for 100Ω transmission line		56		ohms	

Notes

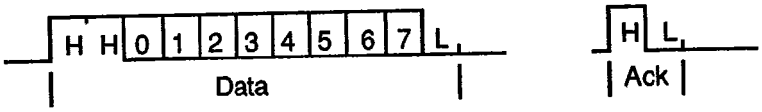
1 This is the variation in the total delay through buffers, transmission lines, differential receivers etc., caused by such things as short term variation in supply voltages and differences in delays for rising and falling edges.



Link Timing



Buffered Link Timing



Link Data and Acknowledge Packets

9 Electrical specifications

9.1 DC electrical characteristics

Absolute Maximum Ratings

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
VCC	DC supply voltage	0		7.0	V	1,2,3
VI, VO	Voltage on input and output pins	-0.5		VCC+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 VCC 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 VCC.

Operating Conditions

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
VCC	DC supply voltage	4.75		5.25	V	1
VI, VO	Input or output voltage	0		VCC	V	1,2
CL	Load capacitance on any pin			50	pF	
TA	Operating temperature range	-55		125	°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.

DC Characteristics

SYMBOL	PARAMETER		MIN	NOM	MAX	UNITS	NOTES
VIH	High level input voltage		2.0		VCC+0.5	V	1,2
VIL	Low level input voltage		-0.5		0.8	V	1,2
II	Input current	@ GND<VI<VCC			±10	μA	1,2
VOH	Output high voltage	@ IOH=2mA	VCC-1			V	1,2
VOL	Output low voltage	@ IOL=4mA			0.4	V	1,2
IOS	Output short circuit current	@ GND<VO<VCC			50	mA	1,2
IOZ	Tristate output current	@ GND<VI<VCC			±10	μA	1,2
PD	Power dissipation				0.7	W	1,2,3
CIN	Input capacitance	@ f=1MHz			7	pF	
COZ	Output capacitance	@ f=1MHz			10	pF	

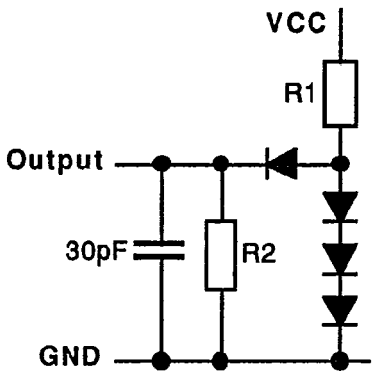
Notes

1 All voltages are with respect to GND.

2 Parameters measured at 4.75V<VCC<5.25V and -55°C<TA<125°C. Input clock frequency = 5MHz.

3 Power dissipation varies with output loading and program execution.

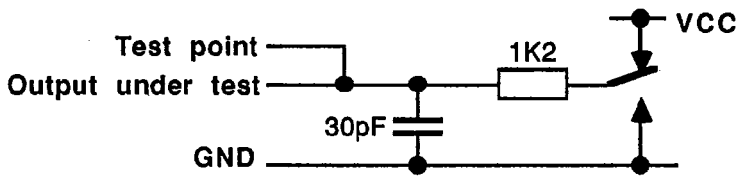
9.2 Equivalent circuits



Load for:	R1	R2	Equivalent load:
Link outputs	1k96	47k	1 Schottky TTL input
Other outputs	970R	24k	2 Schottky TTL inputs

Diodes are 1N916

Load Circuit for AC Measurements



Tristate Load Circuit for AC Measurements

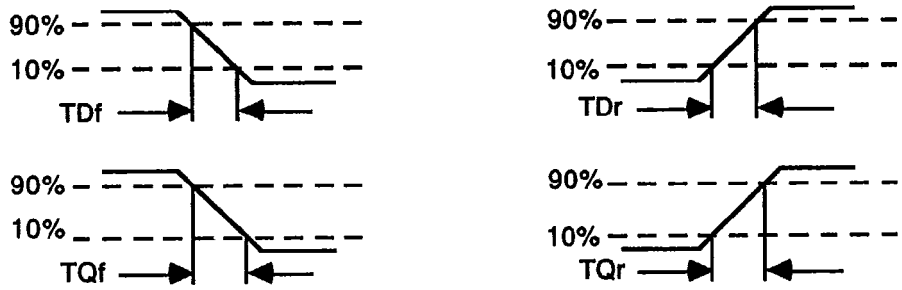
9.3 AC timing characteristics

Input, Output Edges

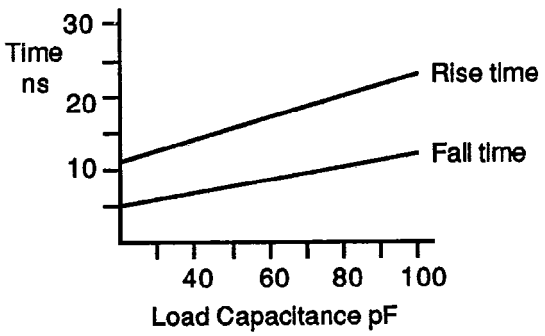
SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
TDr	Input rising edges	2		20	ns	1
TDf	Input falling edges	2		20	ns	1
TQr	Output rising edges			25	ns	1
TQf	Output falling edges			15	ns	1

Notes

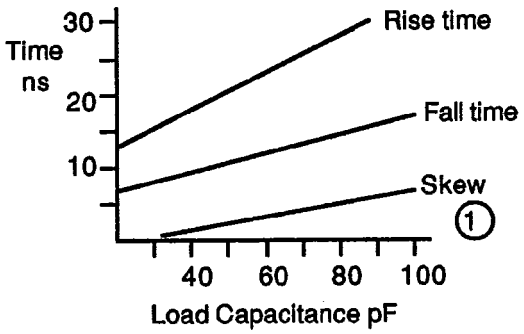
1 Non-link pins; for link timings see section 8.



Input and Output Edge Timing



Typical Link Rise/Fall Times



Typical EMI Rise/Fall Times

Notes

1 Skew is measured between notMemCE with a standard load (2 Schottky TTL inputs and 30pF) and notMemCE with a load of 2 Schottky TTL inputs and varying capacitance.

9.4 Power rating

Internal power dissipation $PINT$ of the chip depends on VCC, as shown in the power dissipation graph. $PINT$ is substantially independent of temperature.

Total power dissipation PD of the chip is

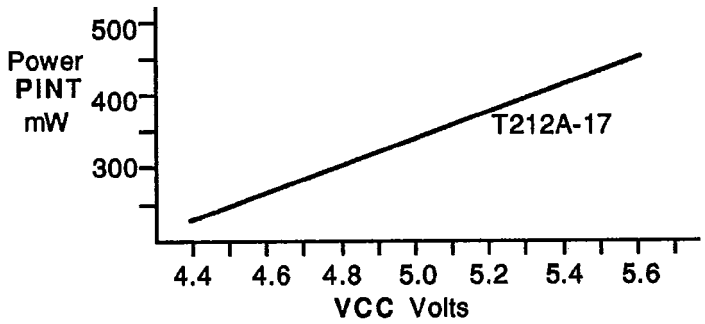
$$PD = PINT + PIO$$

where PIO is the power dissipation in the input and output pins; this is application dependent.

Internal working temperature TJ of the chip is

$$TJ = TA + \theta JA * PD$$

where TA is the external ambient temperature in °C and θJA is the junction-to-ambient thermal resistance in °C/W. θJA for each package is given in the Packaging Specifications section.

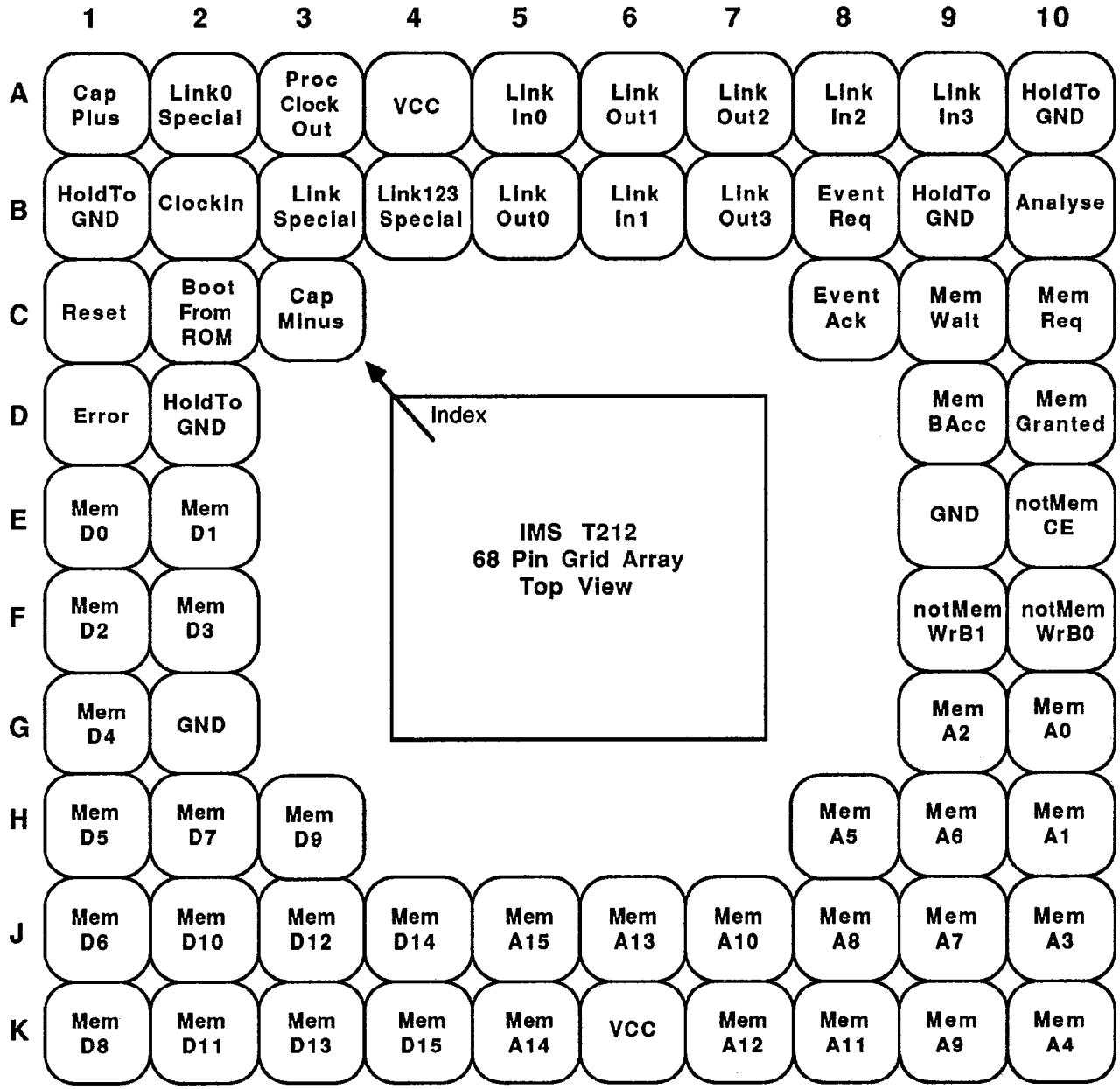


Internal Power Dissipation Vs VCC

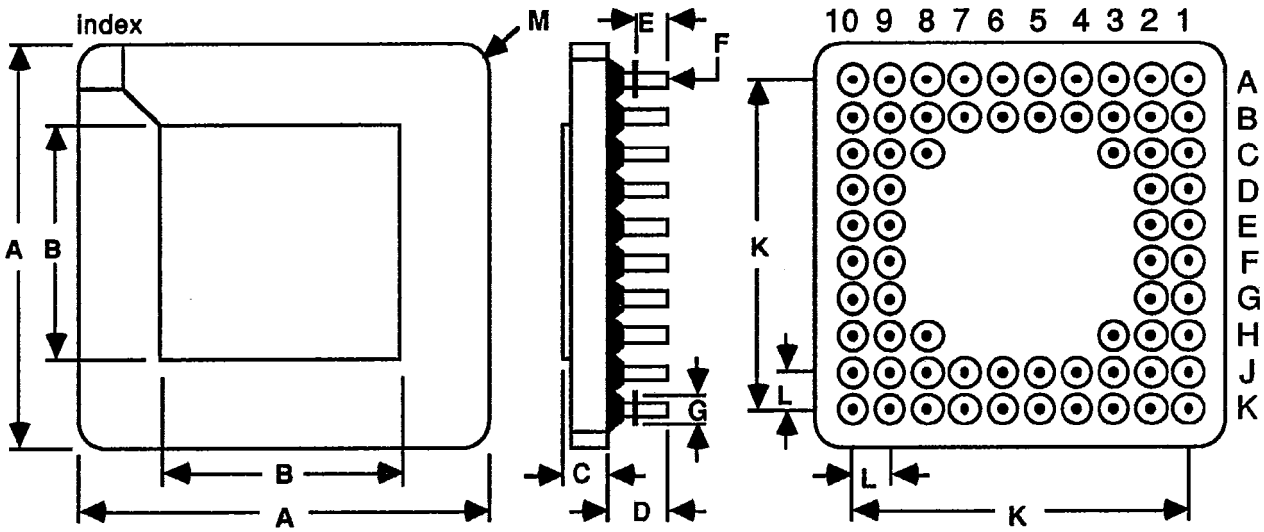
10 Package specifications

10.1 Pin grid array package

10.1.1 Pin grid array pinout



10.1.2 Pin grid array dimensions



DIM	Millimetres		Inches		NOTES
	NOM	TOL	NOM	TOL	
A	26.924	±0.254	1.060	±0.010	
B	17.019	±0.127	0.670	±0.008	
C	2.466	±0.279	0.097	±0.011	
D	4.572	±0.127	0.180	±0.005	
E	3.302	±0.127	0.130	±0.005	
F	0.457	±0.051	0.018	±0.002	Pin diameter
G	1.270	±0.127	0.050	±0.005	Flange diameter
K	22.860	±0.127	0.900	±0.005	
L	2.540	±0.127	0.100	±0.005	
M	0.508		0.020		Chamfer

Package weight is approximately 6.8 grams

10.1.3 Pin grid array thermal characteristics

SYMBOL	PARAMETER	MIN	NOM	MAX	UNITS	NOTES
θJA	Junction to ambient thermal resistance			35	°C/W	1

Notes

1 Measured in still air.

11 Military standard program ‡

The INMOS military program is designed to provide class B microcircuits in accordance with 1.2.1 of MIL-STD-883, "Provisions for the use of MIL-STD-883 in conjunction with compliant non-JAN devices". The IMS T212M is processed for general applications where component quality and reliability must conform to the guidelines and objectives of military procurement. Suitability for use in specific applications should be determined using the guidelines of MIL-STD-454.

Screening procedures are compliant with Method 5004 and the provisions of paragraph 3.3 therein. Quality conformance procedures are compliant with Method 5005 using the alternate Group B provisions of paragraph 3.5.2. All electrical testing is performed to guarantee operation at -55°C, +25°Cand +125°C.

All INMOS military grade components are hermetically sealed in metal-to-ceramic packages.

By specifying an INMOS military product, the user can be assured of receiving a product manufactured, tested and inspected in compliance with MIL-STD-883 and one with superior performance for those applications where quality and reliability are of the essence.

100 Percent Process Step	MIL-STD-883C Method	Test Condition	Comment
Internal visual	2010	B	
Stabilization bake	1008	C	
Temperature cycle	1010	C	
Constant acceleration	2001	D	Y-1 axis
Seal test	1014	B	
Seal test	1014	C	
Visual inspection			INMOS 80-1001
Pre burn-in electrical			+25°C data sheet
Burn-in	1015	D	
Post burn-in electrical			+25°C data sheet
PDA			5% max
Final electrical			+125°C data sheet
Final electrical			-55°C data sheet
External visual	2009		
Group A	5005	3.5.1	A1-A11
Group B	5005	3.5.2	
Group C	5005		MIL-STD-883C 1.2.1.b.17
Group D	5005		MIL-STD-883C 1.2.1.b.17

‡ See INMOS document 41-9047 "Military General Processing Specification" for full details.

12 Ordering details

The following table indicates the designation of the IMS T212M speed and package selections. Speed of **ClockIn** is 5MHz for all parts. Processor cycle time is nominal; it can be calculated more exactly using the phase lock loop factor **PLLx**, as detailed in section 6.1.

INMOS designation	Instruction throughput	Processor clock speed	Processor cycle time	PLLx	Package
IMS T212A-G17M	8.75 MIPS	17.5 MHz	57 ns	3.5	Ceramic Pin Grid