SGS-THOMSON MICROELECTRONICS

IMS T225

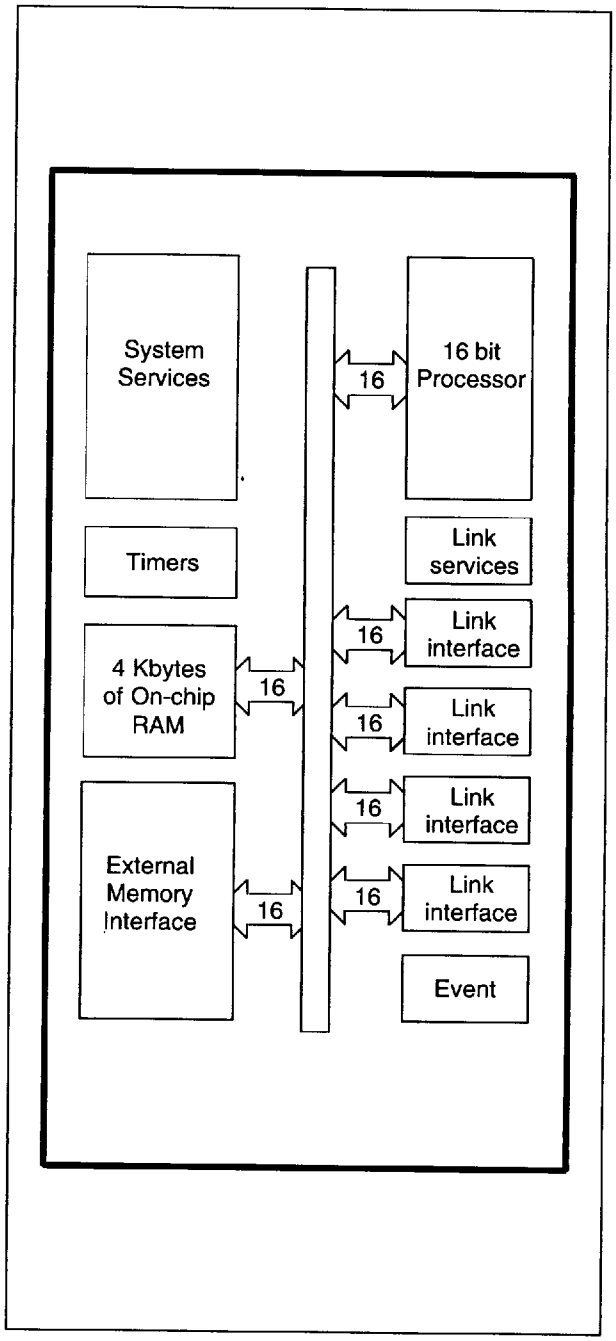
16-bit transputer

FEATURES

- 16 bit architecture
- 33 ns internal cycle time
- 30 MIPS peak instruction rate
- Debugging support
- 4 Kbytes on-chip static RAM
- 60 Mbytes/sec sustained data rate to internal memory
- 64 Kbytes directly addressable external memory
- 30 Mbytes/sec sustained data rate to external memory
- 630 ns response to interrupts
- Four INMOS serial links 5/10/20 Mbits/sec
- Bi-directional data rate of 2.4 Mbytes/sec per link
- Internal timers of 1μs and 64μs
- Boot from ROM or communication links
- Single 5 MHz clock input
- Single +5V ±5% power supply
- Packaging 68 pin PGA / 68 pin PLCC / 100 pin CQFP
- Extended temperature version available

APPLICATIONS

- Real time processing
- Microprocessor applications
- High speed multi processor systems
- Industrial control
- Robotics
- System simulation
- Digital signal processing
- Telecommunications
- Fault tolerant systems
- Medical instrumentation



Contents

1 Introduction

The IMS T225 transputer is a 16 bit CMOS microcomputer with 4 Kbytes on-chip RAM for high speed processing, an external memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages such as ANSI C and provides direct support for concurrency when using either a single transputer or a network. Procedure calls, process switching and typical interrupt latency are sub-microsecond.

For convenience of description, the IMS T225 operation is split into the basic blocks shown in figure 1.1.

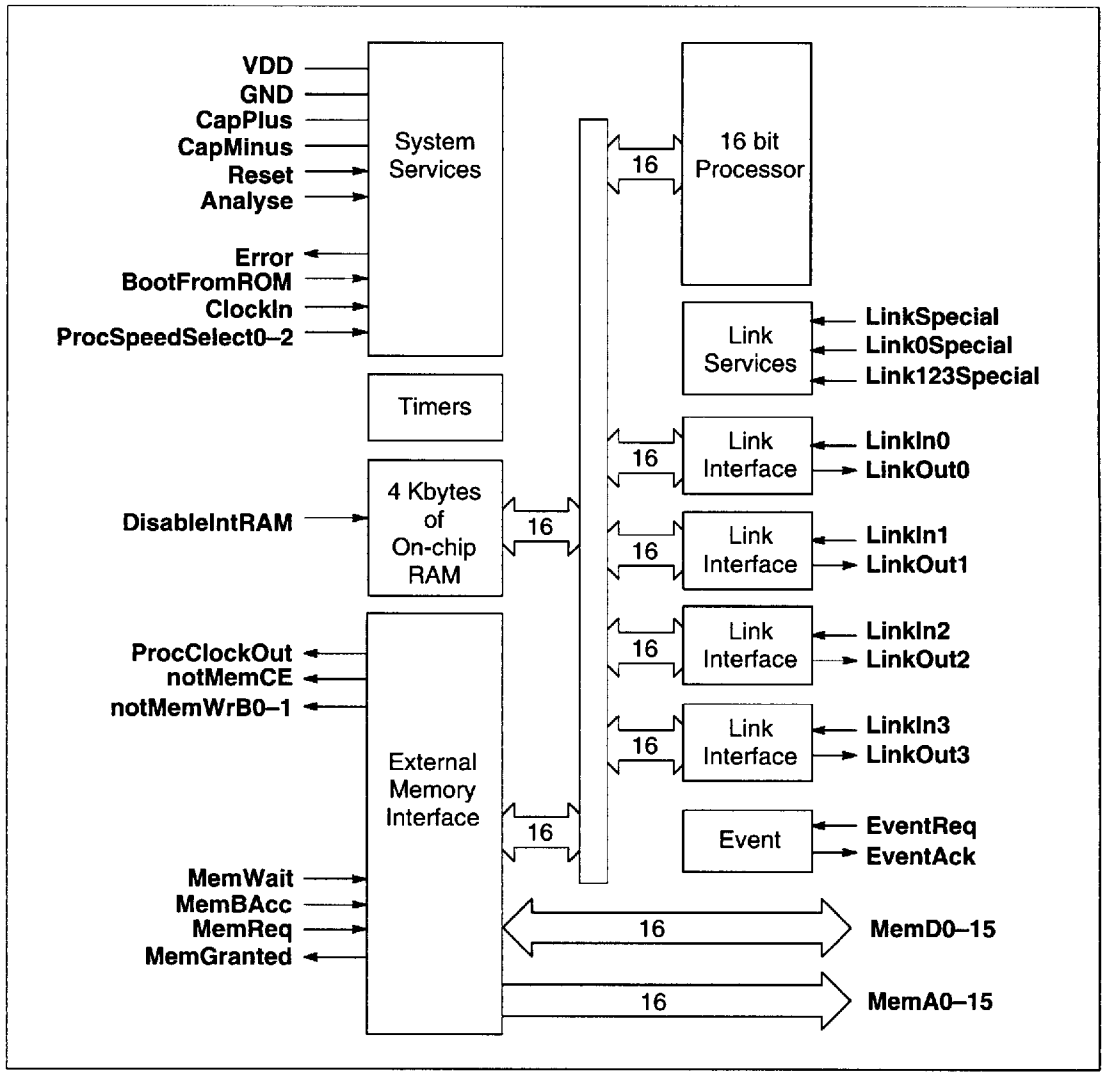


Figure 1.1 IMS T225 block diagram

The IMS T225 is functionally equivalent to the IMS T222 but has the addition of three speed select pins (**ProcSpeedSelect0-2**) and improved links. The IMS T225 is pin compatible with the IMS T222 and is a direct replacement in many applications. The IMS T225 can directly access a linear address space of 64 Kbytes.

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

The INMOS communication links allow networks of transputers to be constructed by direct point to point connections with no external logic. The links support the standard operating speed of 10 Mbits/sec, but

also operate at 5 or 20 Mbits/sec. The links have been improved over those of the IMS T222 and fully support overlapped acknowledge; each IMS T225 link can transfer data bi-directionally at up to 2.4 Mbytes/sec.

The transputer is designed to efficiently implement high level languages such as ANSI C and occam. Access to the transputer at machine level is seldom required, but if necessary refer to the *Transputer Instruction Set – A Compiler Writer's Guide*. A summary of the transputer instruction set can be found in section 11.

The IMS T225 instruction set contains a number of instructions to facilitate the implementation of breakpoints. For further information concerning breakpointing, refer to *Support for debugging/breakpointing in transputers* (technical note 61).

2 Pin designations

Signal names are prefixed by **not** if they are active low, otherwise they are active high.
Pinout details for various packages are given in section 9.

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

Table 2.1 IMS T225 system services

Pin	In/Out	Function
ProcClockOut	out	Processor clock
MemA0–15	out	Sixteen address lines
MemD0–15	in/out	Sixteen data lines
notMemWrB0–1	out	Two byte-addressing write strobes
notMemCE	out	Chip enable
MemBAcc	in	Byte access mode selector
MemWait	in	Memory cycle extender
MemReq	in	Direct memory access request
MemGranted	out	Direct memory access granted

Table 2.2 IMS T225 external memory interface

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

Table 2.3 IMS T225 event

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

Table 2.4 IMS T225 link

3 System services

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

3.1 Power

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

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

3.2 CapPlus, CapMinus

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

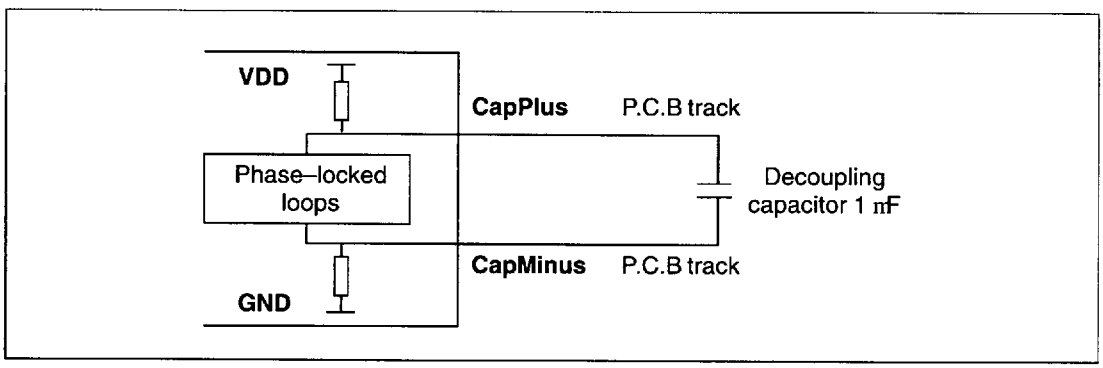


Figure 3.1 Recommended PLL decoupling

3.3 ClockIn

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

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

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

Symbol	Parameter	T225-25			Units	Notes
		Min	Nom	Max		
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** (table 8.3).

Table 3.1 **ClockIn** timing

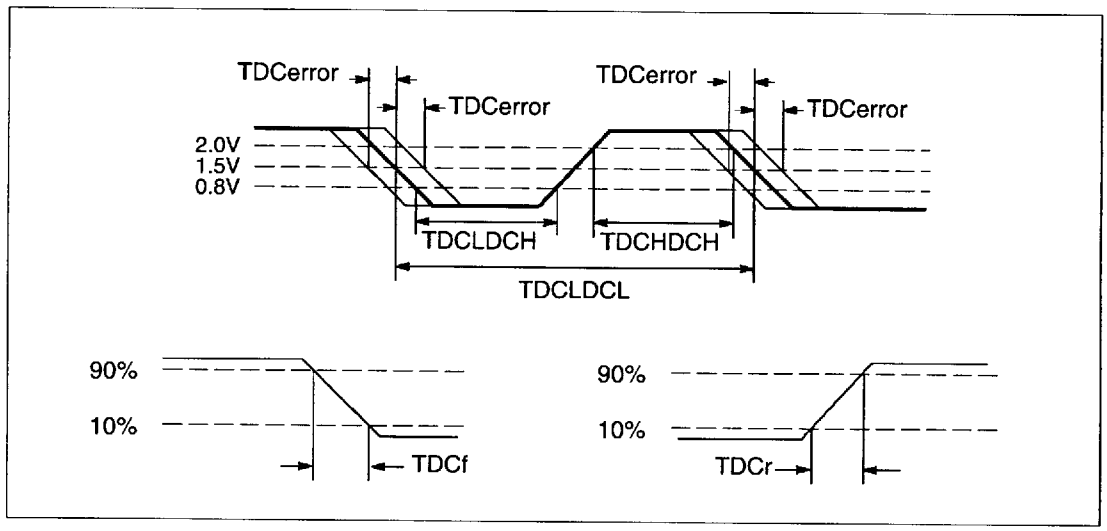


Figure 3.2 **ClockIn** timing

3.4 ProcSpeedSelect0–2

Processor speed of the IMS T225 is variable in discrete steps. The desired speed can be selected, up to the maximum rated for a particular component, by the three speed select lines **ProcSpeedSelect0-2**. The pins are tied high or low, according to the table below, for the various speeds. The pins are arranged so that the IMS T225 can be plugged directly into a board designed for a IMS T222.

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

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

Table 3.2 Processor speed selection

3.5 Bootstrap

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

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the **W** register points to *MemStart* (page 11).

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

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

3.6 Peek and poke

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

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

3.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initializes the transputer and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) must not occur whilst **Reset** is high but can occur before bootstrap (page 23). If **BootFromROM** is high bootstrapping will take place immediately after **Reset** goes low, using data from external memory; otherwise the transputer will await an input from any link. The processor will be in the low priority state.

3.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 46). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags **Error** and **HaltOnError** are not altered at reset, whether **Analyse** is asserted or not.

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

Reset should not be asserted before the transputer has halted and link transfers have ceased. If **BootFromROM** is high the transputer will bootstrap as soon as **Analyse** is taken low, otherwise it will await a control byte on any link. If **Analyse** is taken low without **Reset** going high the transputer state and operation are undefined. After the end of a valid **Analyse** sequence the registers have the values given in table 3.3.

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

Table 3.3 Register values after **Analyse**

Symbol	Parameter	T225-25			Units	Notes
		Min	Nom	Max		
TPVRH	Power valid before **Reset**	10			ms	
TRHRL	**Reset** pulse width high	8			ClockIn	1
TDCVRL	**ClockIn** running before **Reset** end	10			ms	2
TAHRH	**Analyse** setup before **Reset**	3			ms	
TRLAL	**Analyse** hold after **Reset** end	1			ClockIn	1
TBRVRL	**BootFromROM** setup	0			ms	
TRLBRX	**BootFromROM** hold after **Reset**	50			ms	3
TALBRX	**BootFromROM** hold after **Analyse**	50			ms	3

Notes

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

2 At power-on reset.

3 Must be stable until after end of bootstrap period. See Bootstrap section 3.5.

Table 3.4 **Reset** , **Analyse** and **BootFromROM** timing

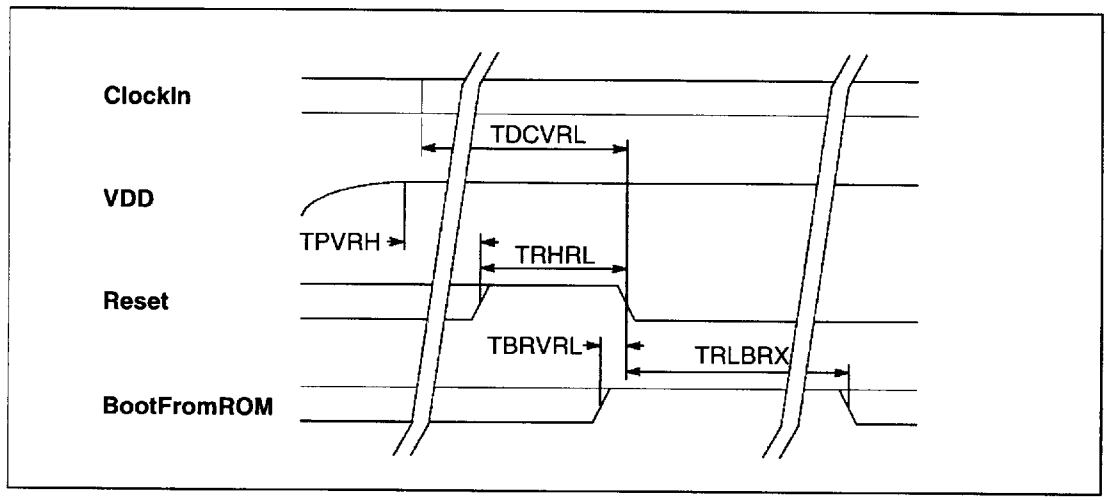


Figure 3.3 Transputer **Reset** timing with **Analyse** low

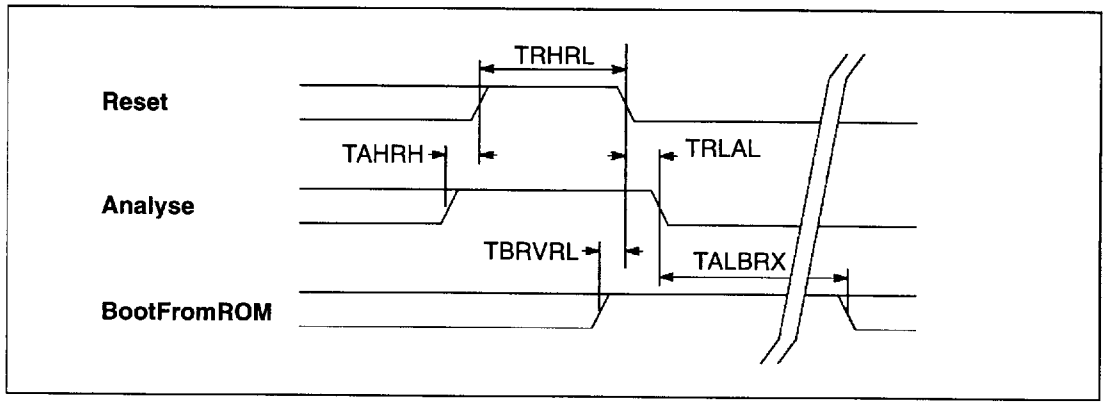


Figure 3.4 Transputer **Reset**, **Analyse** and **BootFromROM** timing

3.9 Error

The **Error** pin is connected directly to the internal *Error* flag and follows the state of that flag. If **Error** is high it indicates an error in one of the processes caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 46). Once set, the *Error* flag is only cleared by executing the instruction *testerr*. The error is not cleared by processor reset, in order that analysis can identify any errant transputer (page 8).

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

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

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

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

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

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

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

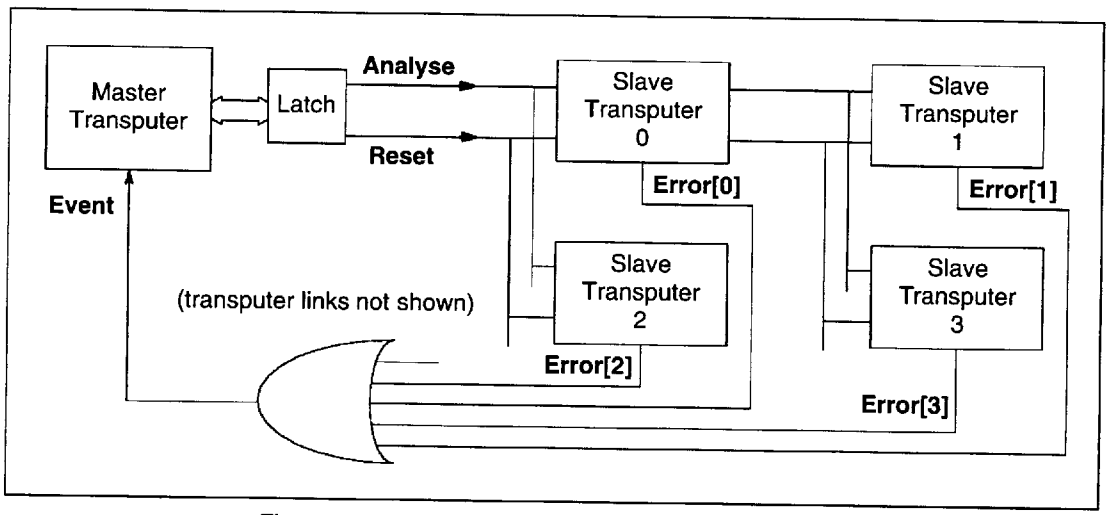


Figure 3.5 Error handling in a multi-transputer system

4 Memory

The IMS T225 has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut**. The transputer can also access an additional 60 Kbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

IMS T225 memory is byte addressed, with words aligned on two-byte boundaries. The least significant byte of a word is the lowest addressed byte.

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

Internal memory starts at the most negative address #8000 and extends to #8FFF. User memory begins at #8024; this location is given the name *MemStart*. An instruction *ldmemstartval* is provided to obtain the value of **MemStart**.

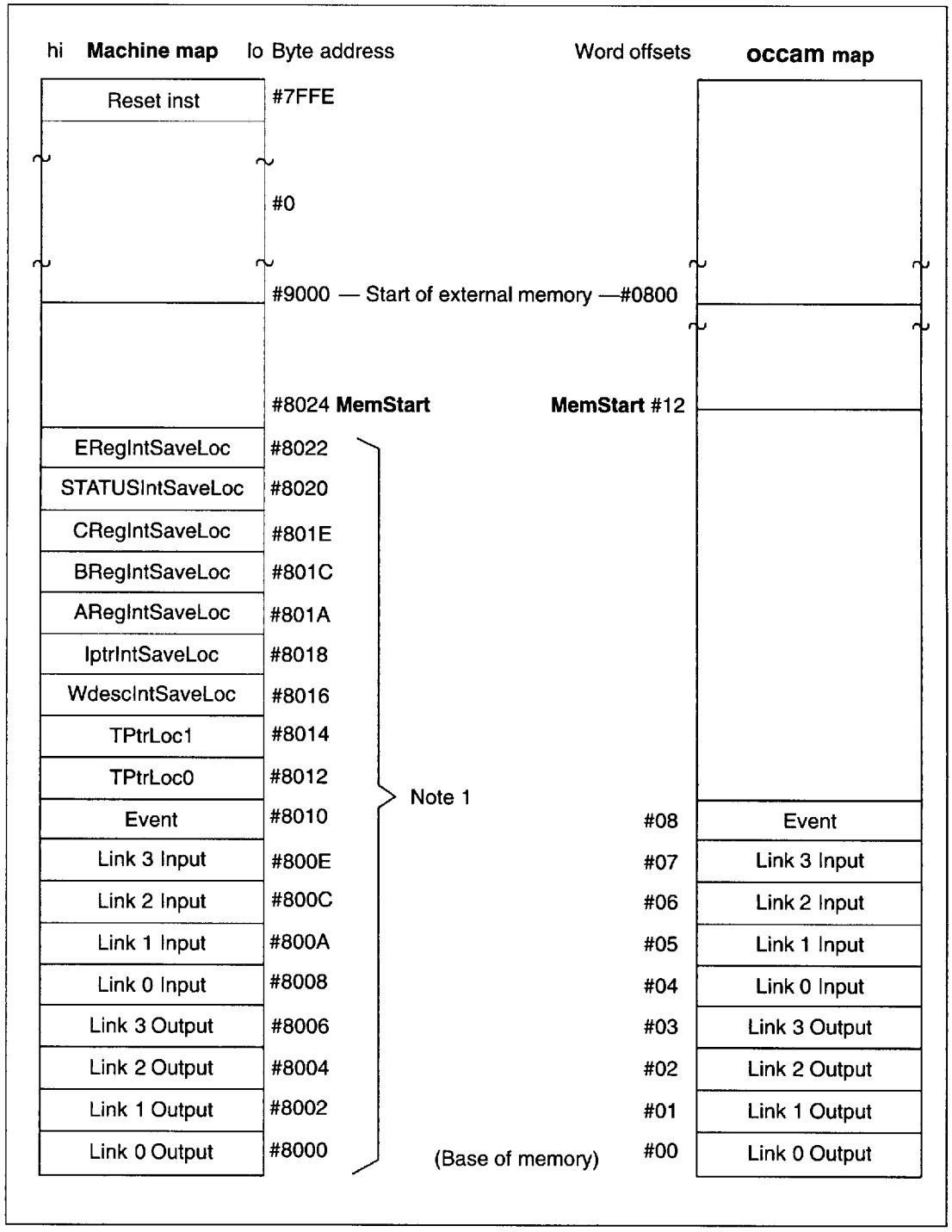
The context of a process in the transputer model involves a workspace descriptor (**WPtr**) and an instruction pointer (**IPtr**). **WPtr** is a word address pointer to a workspace in memory. **IPtr** points to the next instruction to be executed for the process which is the currently executing process.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

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

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

Values of certain processor registers for the current low priority process are saved in the reserved *IntSaveLoc* locations when a high priority process pre-empts a low priority one.

External memory space starts at #9000 and extends up through #0000 to #7FFF. ROM bootstrapping code must be in the most positive address space, starting at #7FFE. Address space immediately below this is conventionally used for ROM based code.



Notes

1 These locations are used as auxiliary processor registers and should not be manipulated by the user. Like processor registers, their contents may be useful for implementing debugging tools (**Analyse**, page 8). For details see the *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 4.1 IMS T225 memory map

5 External memory interface

The IMS T225 External Memory Interface (EMI) can access a 64 Kbyte physical address space, and provides a sustained bandwidth of 30 Mbytes/sec. It accesses a 16 bit wide address space via separate address and data buses. The data bus can be configured for either 16 bit or 8 bit memory access, allowing the use of a single bank of byte-wide memory. Both word-wide and byte-wide access may be mixed in a single memory system (see section 5.5).

The timing parameters given in this chapter are based on tests on a limited number of samples and may change when full characterization is completed.

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

T1 Address and control setup time.

T2 Data setup time.

T3 Data read/write.

T4 Data and address hold after access.

Each **Tstate** is half a processor cycle **TPCLPCL** long (see section 5.2). An external memory cycle is always a complete number of cycles **TPCLPCL** in length and the start of **T1** always coincides with a rising edge of **ProcClockOut**. **T2** can be extended indefinitely by adding externally generated wait states of one complete processor cycle each.

During an internal memory access cycle the external memory interface address bus **MemA0-15** reflects the word address used to access internal RAM, **notMemWrB0-1** and **notMemCE** are inactive and the data bus **MemD0-15** is tristated. This is true unless and until a DMA (memory request) activity takes place, when the lines will be placed in a high impedance state by the transputer.

Bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 8).

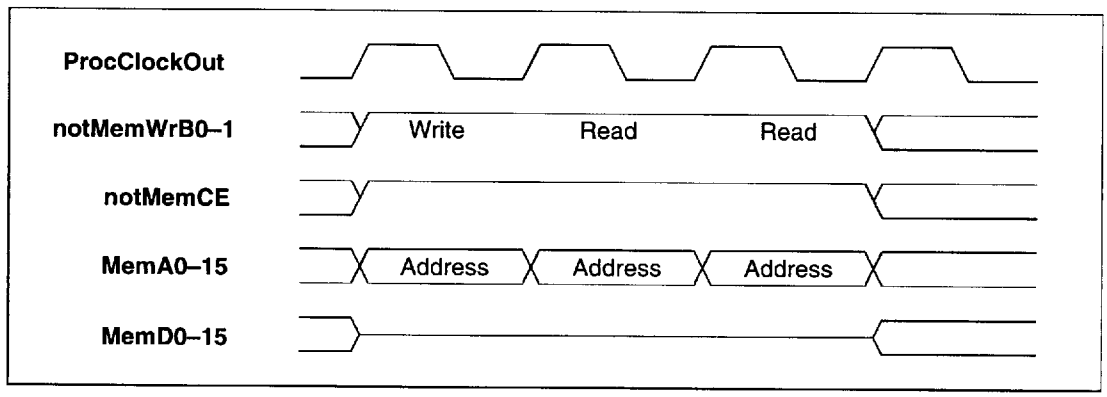


Figure 5.1 IMS T225 bus activity for 3 internal memory cycles

Figure 5.2 below shows an example of an IMS T225 being used in a static RAM application.

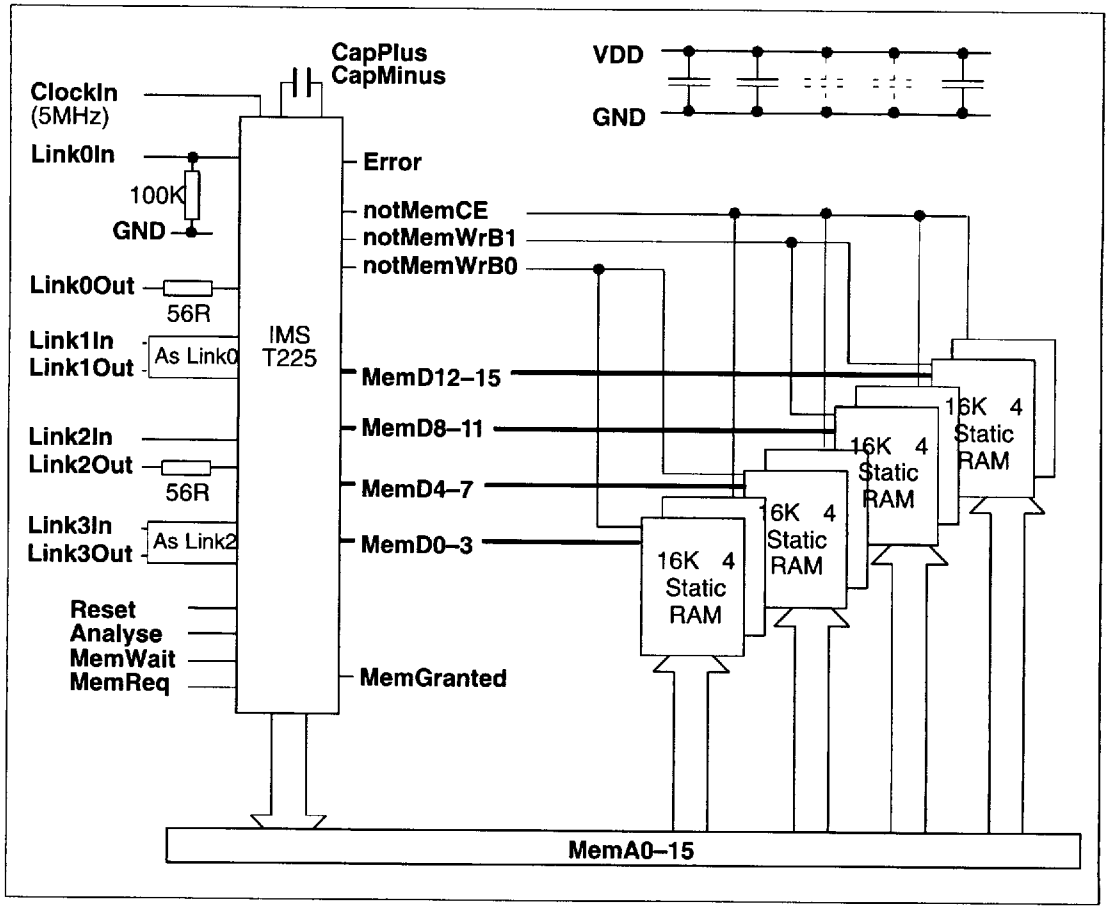


Figure 5.2 IMS T225 static RAM application

5.1 Pin functions

5.1.1 MemA0–15

External memory addresses are output on a non-multiplexed 16 bit address bus (**MemA0–15**). The address is valid at the start of **T1** and remains so until the end of **T4**. Byte addressing is carried out internally by the IMS T225 for read cycles. For write cycles the relevant bytes in memory are addressed by the write enables **notMemWrB0-1**.

5.1.2 MemD0–15

The non-multiplexed data bus (**MemD0–15**) is 16 bits wide. Read cycle data may be set up on the bus at any time after the start of **T1**, but must be valid when the IMS T225 reads it during **T4**. Data can be removed any time after the rising edge of **notMemCE**, but must be off the bus no later than the middle of **T1**, which allows for bus turn-around time before the data lines are driven at the start of **T2** in a processor write cycle. Write data is placed on the bus at the start of **T2** and removed at the end of **T4**. The writing of data into memory is normally synchronized to **notMemCE** going high.

The data bus is high impedance except when the transputer is writing data. If only one byte is being written, the unused 8 bits of the bus are high impedance at that time.

5.1.3 notMemWrB0–1

Two write enables are provided, one to write each byte of the word. When writing a word, both write enables are asserted; when writing a byte only the appropriate write enable is asserted. **notMemWrB0** addresses the least significant byte.

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

Data may be strobed into memory using **notMemWrB0-1** without the use of **notMemCE**, as the write enables go high between consecutive external memory write cycles. The write enables are placed in a high impedance state during DMA, and are inactive during internal memory access.

5.1.4 notMemCE

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

5.1.5 MemBAcc

The IMS T225 performs word access at even memory locations. Access to byte-wide memory can be achieved by taking **MemBAcc** high. Where all external memory operations are to byte-wide memory, **MemBAcc** may be wired permanently high. The state of this signal is latched during **T2**.

If **MemBAcc** is low then a full word will be accessed in one external memory cycle, otherwise the high and low bytes of the word will be separately accessed during two consecutive cycles. The first (least significant) byte is accessed at the word address (**MemA0** is low). The second (most significant) byte is accessed at the word address +1 (**MemA0** is high).

5.1.6 MemWait

If the data setup time for read or write is too short it can be extended by inserting wait states at the end of **T2** (see section 5.6). Wait states can be selected by taking **MemWait** high. **MemWait** is sampled during **T2**, and should not change state in this region.

Internal memory access is unaffected by the number of wait states selected.

5.1.7 MemReq, MemGranted

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. **MemGranted** can be used to signal to the device requesting the DMA that it has control of the bus.

For external memory cycles, the IMS T225 samples **MemReq** during the first high phase of **ProcClockOut** after **notMemCE** goes low. In the absence of an external memory cycle, **MemReq** is sampled during every rising edge of **ProcClockOut**. **MemA0-15**, **MemD0-15**, **notMemWrB0-1** and **notMemCE** are tristated before **MemGranted** is asserted.

5.1.8 ProcClockOut

This clock is derived from the internal processor clock, which is in turn derived from **ClockIn** (see section 5.2).

5.2 Processor clock

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

TPCLPCL = TDCLDCL / PLLx

where **TPCLPCL** is the **ProcClockOut Period**, **TDCLDCL** is the **ClockIn Period** and **PLLx** is the phase lock loop factor for the relevant speed part, obtained from the ordering details (refer to section 10).

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

Symbol	Parameter	T225-25		Units	Notes
		Min	Max		
TPCLPCL	**ProcClockOut** period	38	42	ns	
TPCHPCL	**ProcClockOut** pulse width high	14	26	ns	
TPCLPCH	**ProcClockOut** pulse width low	**a**		ns	2,3
TPCstab	**ProcClockOut** stability		8	%	1

Notes

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

2 **a** is TPCLPCL – TPCHPCL.

3 This is a nominal value.

Table 5.1 **ProcClockOut**

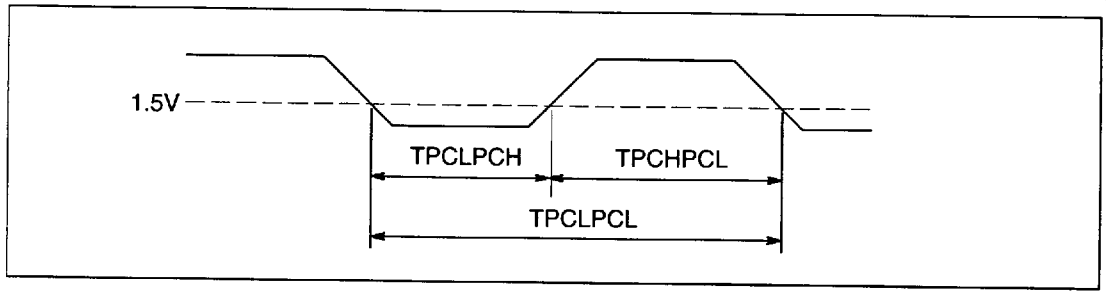


Figure 5.3 IMS T225 **ProcClockOut** timing

5.3 Read cycles

External memory addresses are output on the non-multiplexed 16 bit address bus (**MemA0–15**). The address is valid at the start of **T1** and remains so until the end of **T4**, with the timing shown in figure 5.4. Byte addressing is carried out internally by the IMS T225 for read cycles.

The non-multiplexed data bus (**MemD0–15**) is 16 bits wide. Read cycle data may be set up on the data bus at any time after the start of **T1**, but must be valid when the IMS T225 reads it during **T4**. Data can be removed any time after the rising edge of **notMemCE**, but must be off the bus no later than the middle of **T1**, which allows for bus turn-around time before the data lines are driven at the start of **T2** in a processor write cycle.

Symbol	Parameter	T225-25		Units	Notes
		Min	Max		
TAVEL	Address valid before chip enable low	6		ns	1
TELEH	Chip enable low	54	66	ns	1
TEHEL	Delay before chip enable re-assertion	15		ns	1, 2
TEHAX	Address hold after chip enable high	1		ns	1
TELDrV	Data valid from chip enable low	0	40	ns	
TAVDrV	Data valid from address valid	0	53	ns	
TDrVEH	Data setup before chip enable high	17		ns	
TEHDrZ	Data hold after chip enable high	0		ns	
TWEHEL	Write enable setup before chip enable low	16		ns	3
TPCHEL	**ProcClockOut** high to chip enable low	4	17	ns	1
TEHPCH	Chip enable high to **ProcClockOut** high	5		ns	

Notes

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

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

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

Table 5.2 Read

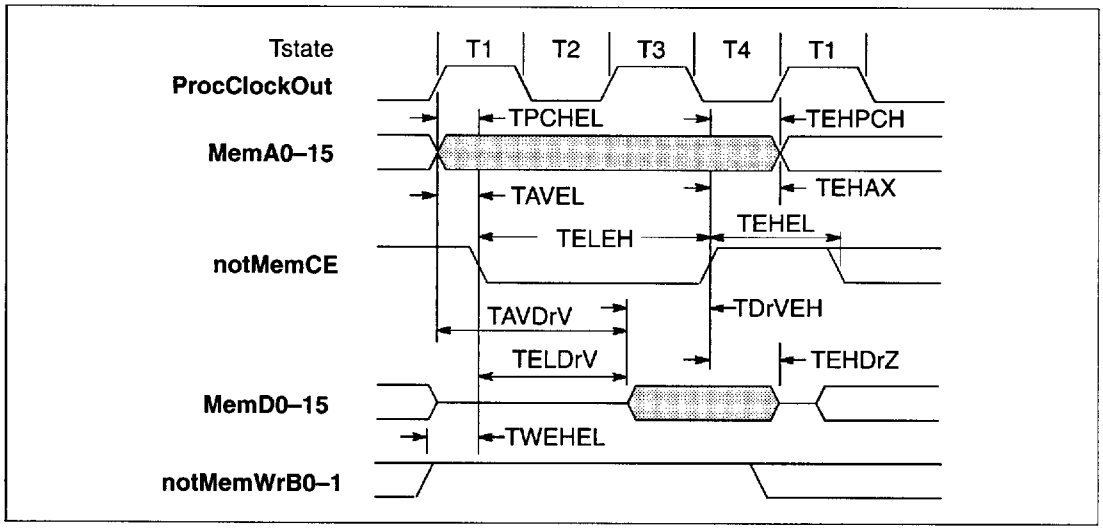


Figure 5.4 IMS T225 external read cycle

5.4 Write cycles

For write cycles the relevant bytes in memory are addressed by the write strobes **notMemWrB0-1**.

Write data is placed on the data bus (**MemD0–15**) at the start of **T2** and removed at the end of **T4**. It is normally written into memory in synchronism with **notMemCE** going high.

Two write strobes are provided, one to write each byte of the word. When writing a word, both write strobes are asserted; when writing a byte only the appropriate write enable is asserted. **notMemWrB0** addresses the least significant byte.

The IMS T225 will, by default, perform word access at even memory locations, this is termed word access mode. Access to byte-wide memory can be achieved by taking the **MemBAcc** signal high, this is termed byte access mode (see section 5.5.2). In word access mode a full word will be accessed in one external memory cycle, in byte access mode the high and low bytes of the word will be separately accessed during two consecutive cycles. Both word-wide and byte-wide access may be mixed in a single memory system. Figure 5.6 shows a write access of the least significant byte of the word when in word access mode (**MemBAcc** low). **MemA0** is low to signify access of the least significant byte of the word at the word address.

Data may be strobed into memory using **notMemWrB0-1** without the use of **notMemCE**, as the write strobes go high between consecutive external memory write cycles. The write strobes are placed in a high impedance state during DMA, and are inactive during internal memory access.

Symbol	Parameter	T225-25		Units	Notes
		Min	Max		
TDwVEH	Data setup before chip enable high	40		ns	
TEHDwZ	Data hold after write	3	20	ns	
TDwZEL	Write data invalid to next chip enable	1		ns	
TWELEL	Write enable setup before chip enable low	–3	3	ns	1
TEHWEH	Write enable hold after chip enable high	–3	3	ns	1

Notes

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

Table 5.3 Write

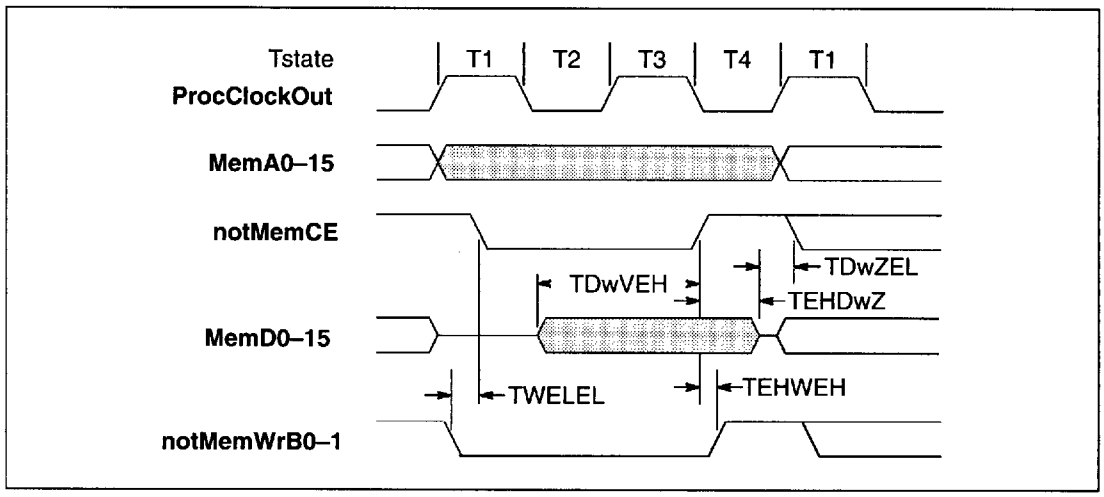


Figure 5.5 IMS T225 external write cycle

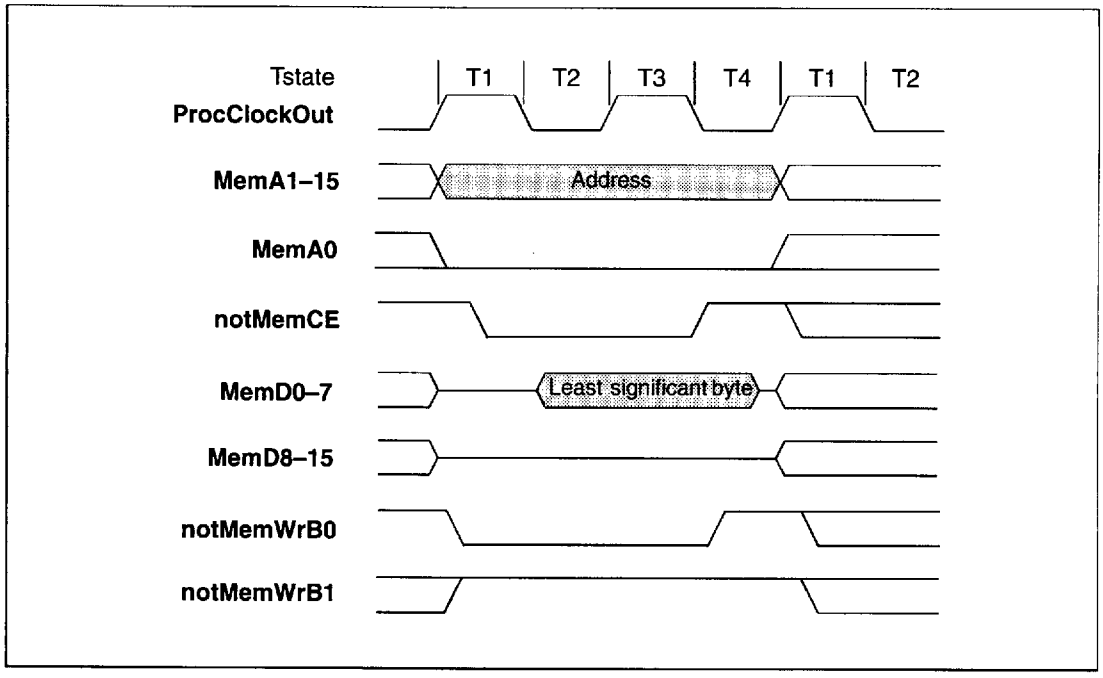


Figure 5.6 IMS T225 least significant byte write in word access mode

5.5 MemBAcc

The IMS T225 will, by default, perform word access at even memory locations. Access to byte-wide memory can be achieved by taking **MemBAcc** high with the timing shown. Where all external memory operations are to byte-wide memory, **MemBAcc** may be wired permanently high. The state of this signal is latched during **T2**. Where external memory operations may be to both byte and word wide memory, **MemBAcc** should be obtained by address decoding. If you use a memory system in which word wide memory may be used in byte access mode it is recommended that **notMemWrB1** is OR gated with **MemA0**, to prevent any spurious data being written to the RAM.

If **MemBAcc** is low then a full word will be accessed in one external memory cycle, otherwise the high and low bytes of the word will be separately accessed during two consecutive cycles. The first (least significant) byte is accessed at the word address (**MemA0** is low). The second (most significant) byte is accessed at the word address +1 (**MemA0** is high).

5.5.1 Word Read/Write in Byte Access Mode

With **MemBAcc** high, the first cycle is identical with a normal word access cycle. However, it will be immediately followed by another memory cycle, which will use **MemD0-7** to read or write the second (most significant) byte of data. During this second cycle, for a write, **notMemWrB0–1** both go low as in the first cycle and **MemA0** goes high. For a read, **notMemWrB0–1** remain high and **MemA0** goes high. **MemD8–15** are high impedance for both read and write in the second cycle. Figure 5.7 shows a word write to byte-wide memory.

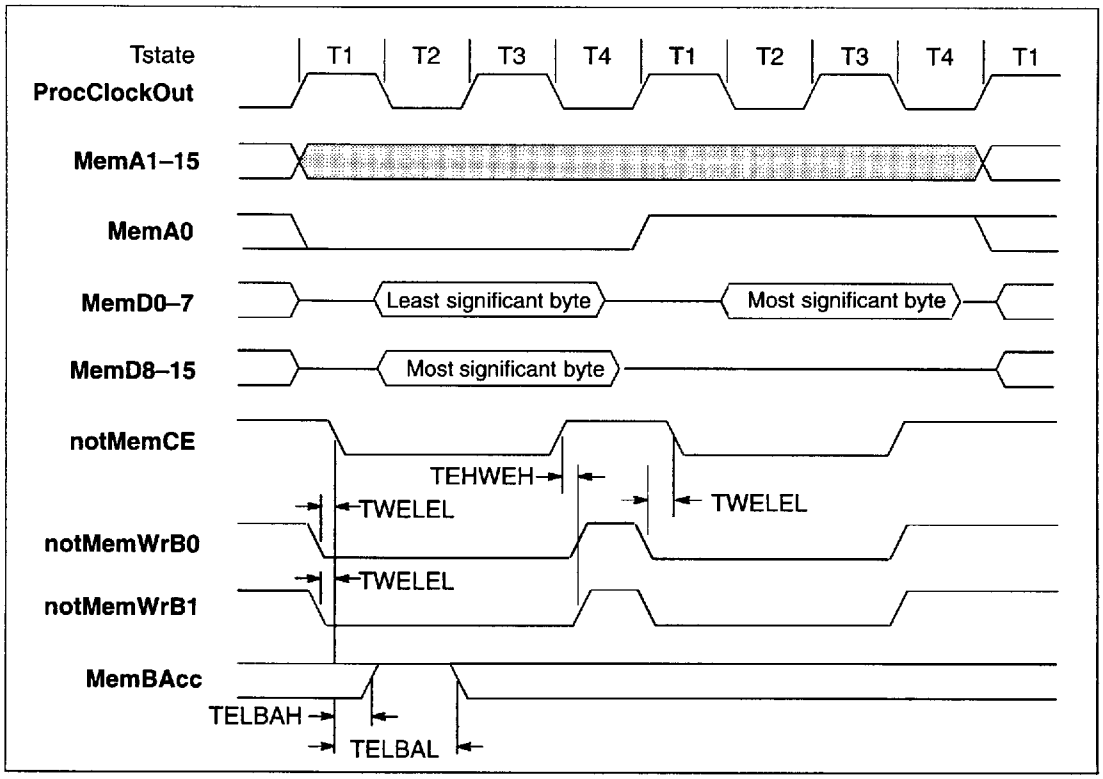


Figure 5.7 IMS T225 word write to byte-wide memory

5.5.2 Byte Write in Byte Access Mode

Writing a Most Significant Byte

In the first cycle **notMemWrB1** will go low and **notMemWrB0** will remain high. **MemA0** remains low. In the second cycle **MemA0** goes high and **notMemWrB0–1** go low. The data is written on **MemD0–7** in the second cycle.

Figure 5.8 shows a write access of the most significant byte of the word when in byte access mode (**MemBAcc** high). During the first access a normal word access is performed with **notMemWrB1** active low to select the most significant byte. During the second cycle, **MemD0-7** writes the most significant byte accessed at word address + 1 (**MemA0** is high).

Writing a Least Significant Byte

In the first cycle **notMemWrB1** remains high and **notMemWrB0** goes low. **MemA0** remains low. In the second cycle **MemA0** and **notMemWrB0** go high, **notMemWrB1** remains high. Data is written on **MemD0–7** in the first cycle.

Symbol	Parameter	T225-25		Units	Notes
		Min	Max		
TELBAH	**MemBAcc** high from chip enable		10	ns	
TELBAL	**MemBAcc** low from chip enable	27		ns	

Table 5.4 Byte-wide memory access

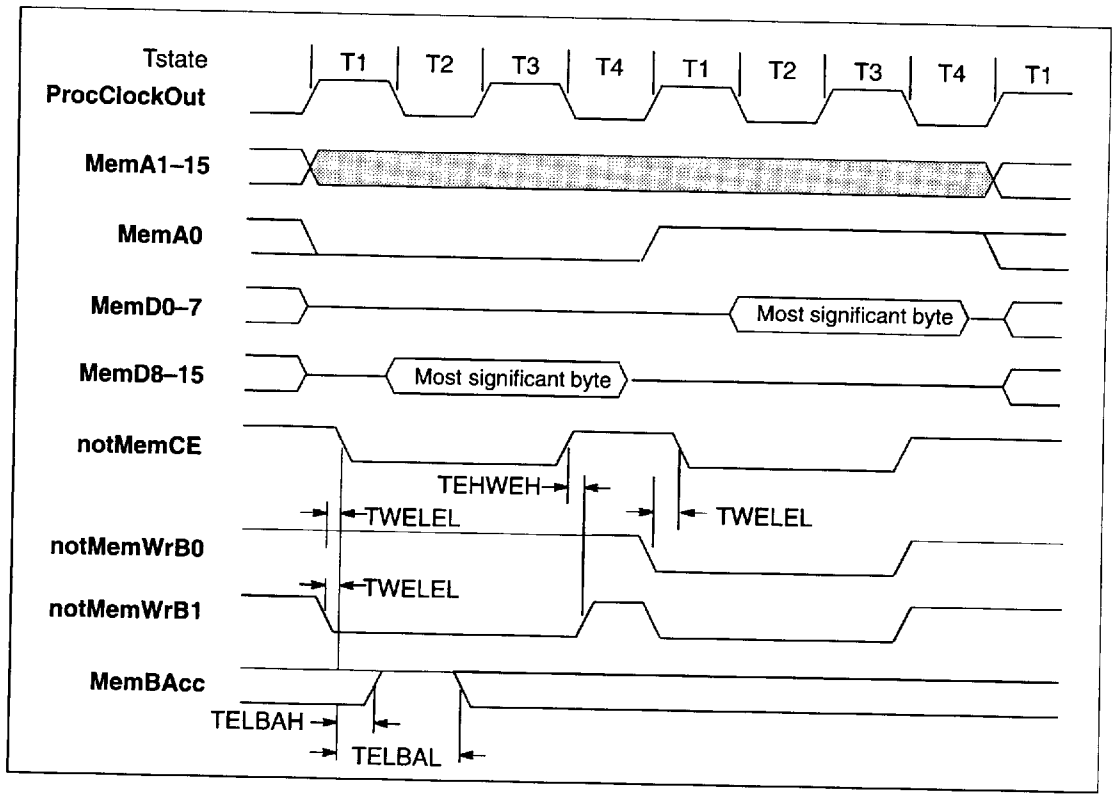


Figure 5.8 IMS T225 most significant byte write to byte-wide memory

5.6 Wait

Wait states can be selected by taking **MemWait** high. **MemWait** is sampled during **T2**, and should not change state in this region.

A wait state is one processor cycle TPCLPCL long and comprises the pair **W1** and **W2**, each half a processor cycle long (see figure 5.9). If **MemWait** is still high when sampled in **W2** then another wait period will be inserted. This can continue indefinitely. Internal memory access is unaffected by the number of wait states selected.

The setup and hold timing requirements for **MemWait** are the same as for a normal word read/write cycle. Each wait state inserted extends the length of **MemA0-15**, **MemD0-7**, **notMemCE**, and **notMemWrB0** by one **ProcClockOut** cycle (TPCLPCL).

If wait states are required to extend a byte access cycle, then the time in the cycle at which **MemBAcc** needs to be asserted is delayed (relative to the falling edge of **notMemCE**) by one TPCLPCL for each wait state inserted (see figure 7.9). **MemWait** also needs to be re-asserted for the second byte accessed (**MemA0** is high), to extend this cycle. The timing requirements of the second assertion of wait are identical to the first except that the timing is relative to the equivalent rising edge of **ProcClockOut** in the second byte access. Note that the number of wait states inserted in the even and odd byte accesses can be different.

Symbol	Parameter	T225-25		Units	Notes
		Min	Max		
TPCHWtH	**MemWait** asserted after **ProcClockOut** high		20	ns	
TPCHWtL	**MemWait** low after **ProcClockOut** high	40		ns	

Table 5.5 Memory wait

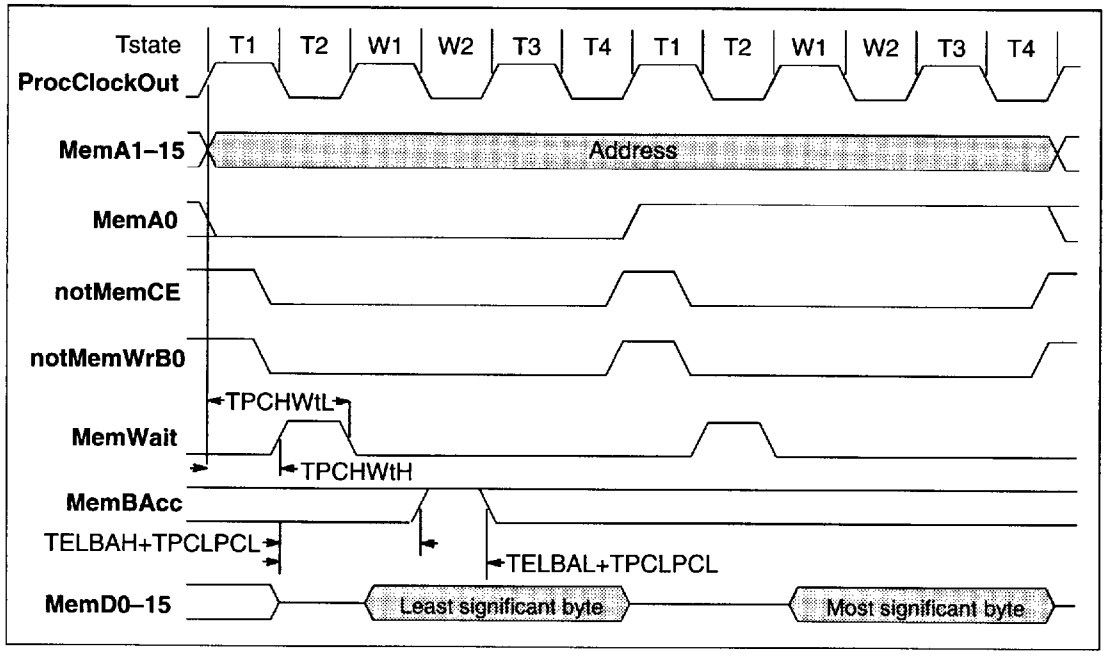


Figure 5.9 IMS T225 word write to byte wide memory with a single wait state

The wait state generator can be a simple digital delay line, synchronized to **notMemCE**. The **Single Wait State Generator** circuit in figure 5.10 can be extended to provide two or more wait states, as shown in figure 5.11.

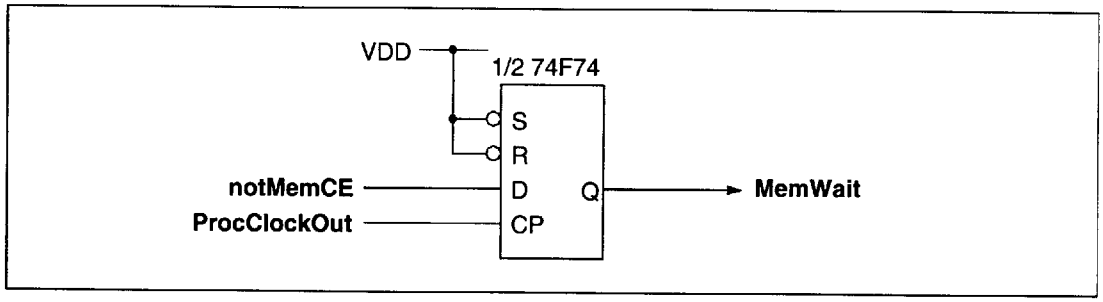


Figure 5.10 Single wait state generator

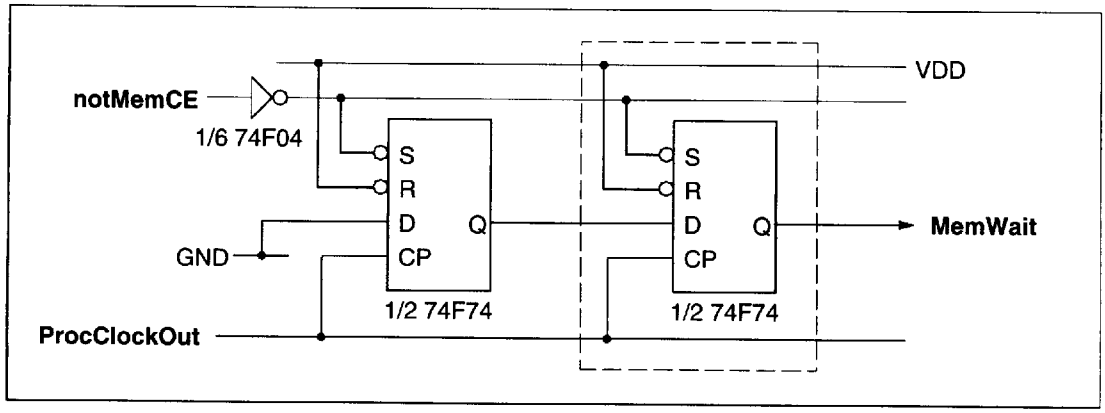


Figure 5.11 Extendable wait state generator

5.7 Direct memory access

Direct memory access (DMA) can be requested at any time by taking the asynchronous **MemReq** input high. For external memory cycles, the IMS T225 samples **MemReq** during the first high phase of **ProcClockOut** after **notMemCE** goes low. In the absence of an external memory cycle, **MemReq** is sampled during every rising edge of **ProcClockOut**. **MemA0-15**, **MemD0-15**, **notMemWrB0-1** and **notMemCE** are tristated before **MemGranted** is asserted.

Removal of **MemReq** is sampled at each rising edge of **ProcClockOut** and **MemGranted** removed with the timing shown in figure 5.14. Further external bus activity, either external cycles or reflection of internal cycles, will commence during the next low phase of **ProcClockOut**.

notMemCE, **notMemWrB0-1**, **MemA0-15** and **MemD0-15** are in a high impedance state during DMA. External circuitry must ensure that **notMemCE** and **notMemWrB0-1** do not become active whilst control is being transferred; it is recommended that a 10K resistor is connected from **VDD** to each pin. DMA cannot interrupt an external memory cycle. DMA does not interfere with internal memory cycles in any way, although a program running in internal memory would have to wait for the end of DMA before accessing external memory. DMA cannot access internal memory.

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

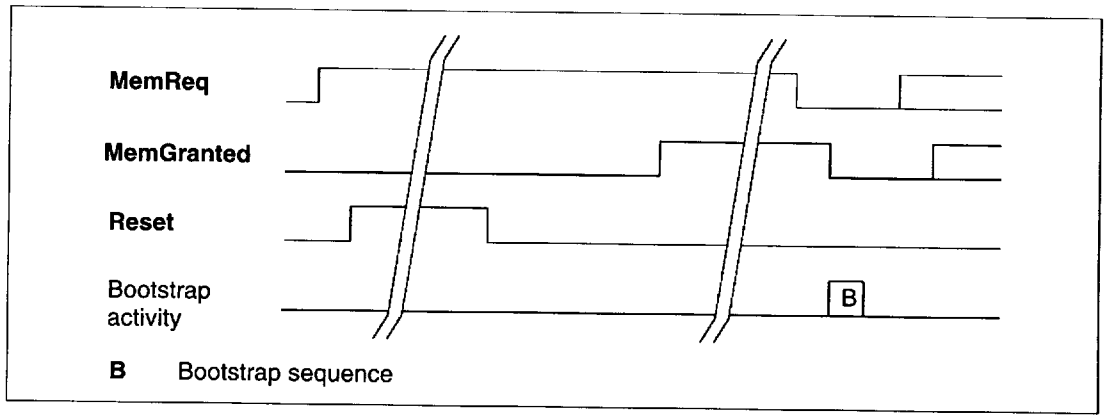


Figure 5.12 IMS T225 DMA sequence at reset

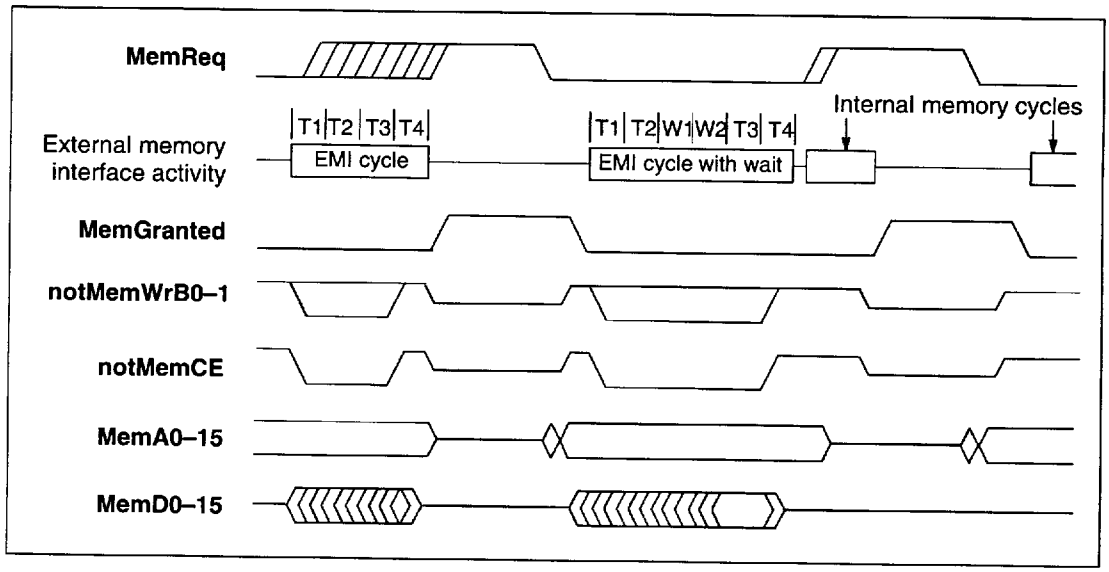


Figure 5.13 IMS T225 operation of **MemReq** and **MemGranted** with external and internal memory cycles

Symbol	Parameter	T225–25		Units	Notes
		Min	Max		
TMRHMGH	Memory request response time	60	**a**	ns	1
TMRLMGL	Memory request end response time	65	125	ns	
TAZMGH	Address bus tristate before **MemGranted**	0		ns	
TAVMGL	Address bus active after **MemGranted** end	0		ns	
TDZMGH	Data bus tristate before **MemGranted**	0		ns	
TEZMGH	Chip enable tristate before **MemGranted**	0		ns	2
TEVMGL	Chip enable active after **MemGranted** end	–6		ns	
TWEZMGH	Write enable tristate before **MemGranted**	0		ns	2
TWEVMGL	Write enable active after **MemGranted** end	–6		ns	

Notes

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

2 When using DMA, **notMemCE** and **notMemWrB0-1** should be pulled up with a resistor (typically 10K). Capacitance should be limited to a maximum of 50pF.

Table 5.6 Memory request timing

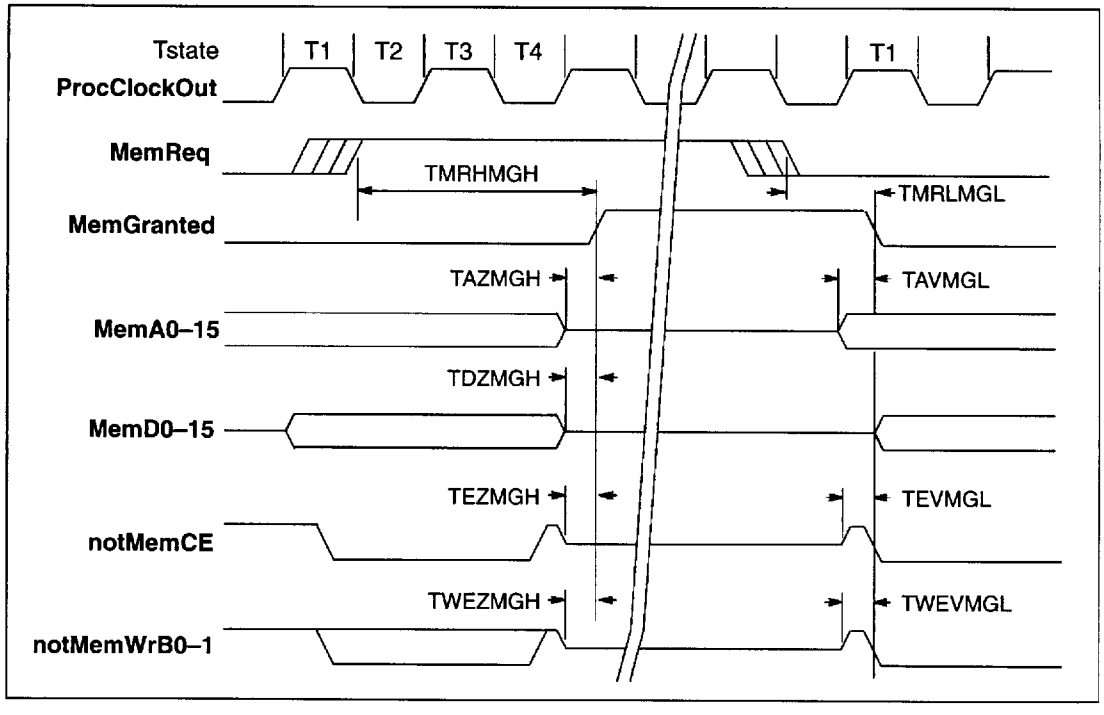


Figure 5.14 IMS T225 memory request timing

6 Events

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

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

If the process is a high priority one and no other high priority process is running, typical latency is 19 full processor cycles **TCPLCPL**, and maximum latency (assuming all memory accesses are internal) is 53 full processor cycles. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

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

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

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

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

Symbol	Parameter	T225-25		Units
		Min	Max	
TVHKH	**EventReq** response	0		ns
TKHVL	**EventReq** hold	0		ns
TVLKL	Delay before removal of **EventAck**	0	127	ns
TKLVH	Delay before re-assertion of **EventReq**	0		ns

Table 6.1 Event

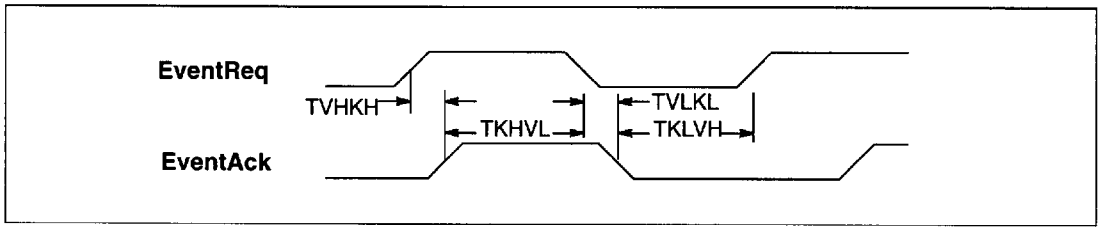


Figure 6.1 IMS T225 event timing

7 Links

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

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

The IMS T225 links support the standard INMOS communication speed of 10 Mbits/sec. In addition they can be used at 5 or 20 Mbits/sec for 25 MHz devices, and 20 Mbits/sec for faster devices. Links are not synchronised with **ClockIn** or **ProcClockOut** and are insensitive to their phases. Thus links from independently clocked systems may communicate, providing only that the clocks are nominally identical and within specification.

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

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

Link speeds can be set by **LinkSpecial**, **Link0Special** and **Link123Special**. The link 0 speed can be set independently. Table 7.1 shows uni-directional and bi-directional data rates in Kbytes/sec for each link speed; **LinknSpecial** is to be read as **Link0Special** when selecting link 0 speed and as **Link123Special** for the others. Data rates are quoted for a transputer using internal memory, and will be affected by a factor depending on the number of external memory accesses and the length of the external memory cycle.

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

Table 7.1 Speed settings for transputer links

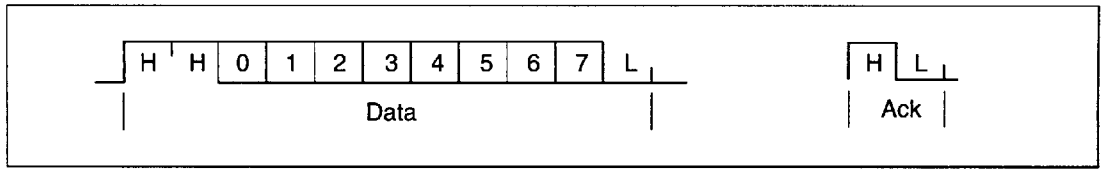


Figure 7.1 IMS T225 link data and acknowledge packets

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	20 Mbits/s			3	ns	1
		10 Mbits/s			10	ns	1
		5 Mbits/s			30	ns	1
CLIZ	**LinkIn** capacitance	@ f=1MHz			7	pF	
CLL	**LinkOut** load capacitance				50	pF	
RM	Series resistor for 100W 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.

Table 7.2 Link

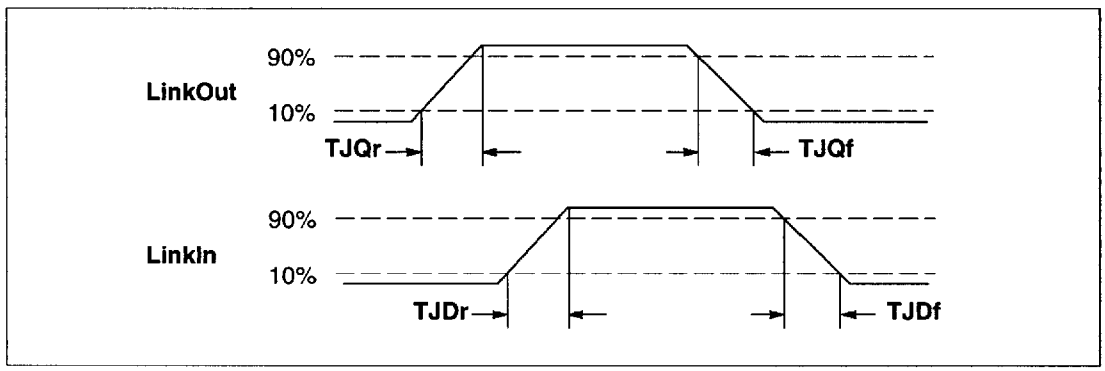


Figure 7.2 IMS T225 link timing

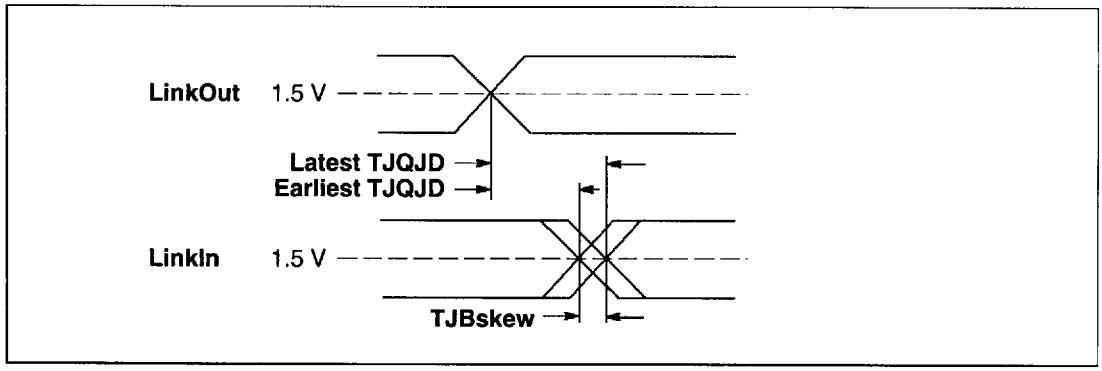


Figure 7.3 IMS T225 buffered link timing

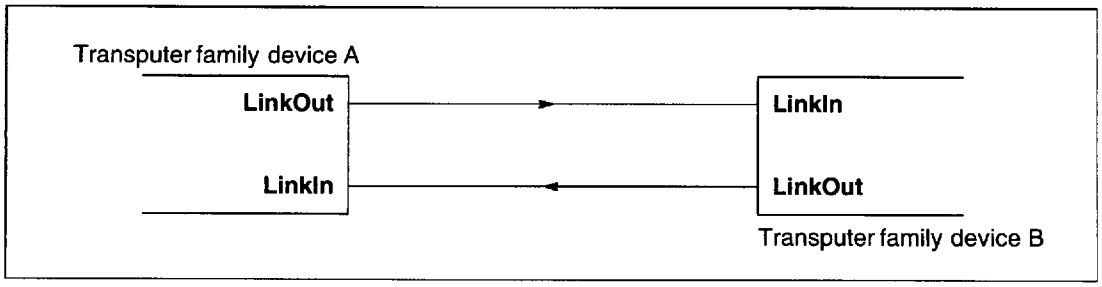


Figure 7.4 IMS T225 links directly connected

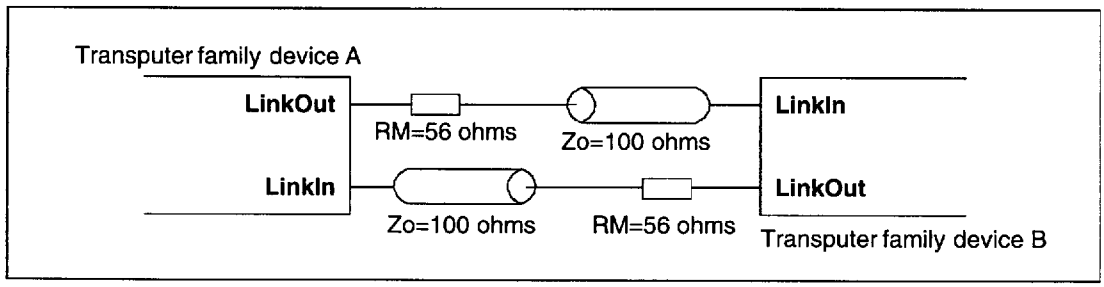


Figure 7.5 IMS T225 links connected by transmission line

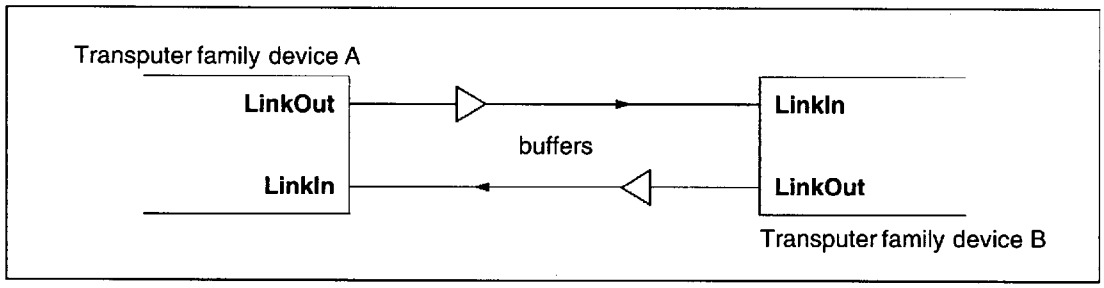


Figure 7.6 IMS T225 links connected by buffers

8 Electrical specifications

8.1 Absolute maximum ratings

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

Notes

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

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

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

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

Table 8.1 Absolute maximum ratings

8.2 Operating conditions

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

Notes

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

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

3 Excluding **LinkOut** load capacitance.

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

Table 8.2 Operating conditions

8.3 DC electrical characteristics

SYMBOL	PARAMETER		MIN	MAX	UNITS	NOTES
VIH	High level input voltage		2.0	VDD+0.5	V	1, 2
VIL	Low level input voltage		–0.5	0.8	V	1, 2
II	Input current	@ GND<VI<VDD		±10	μA	1, 2
VOH	Output high voltage	@ IOH=2mA	VDD–1		V	1, 2
VOL	Output low voltage	@ IOL=4mA		0.4	V	1, 2
IOZ	Tristate output current	@ GND<V0<VDD		±10	μA	1, 2
PD	Power dissipation			700	mW	2, 3
CIN	Input capacitance	@ f=1MHz		7	pF	
COZ	Output capacitance	@ f=1MHz		10	pF	

Notes

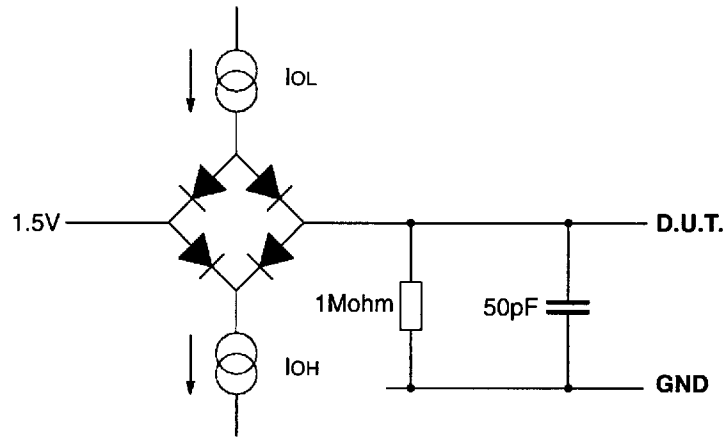
1 All voltages are with respect to **GND**.

2 Parameters for IMS T225-S measured at 4.75V<**VDD**<5.25V and 0°C<**TA**<70°C. Input clock frequency = 5 MHz.

3 Power dissipation varies with output loading and program execution.

Table 8.3 DC characteristics

8.4 Equivalent circuits



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

Figure 8.1 Load circuit for AC measurements

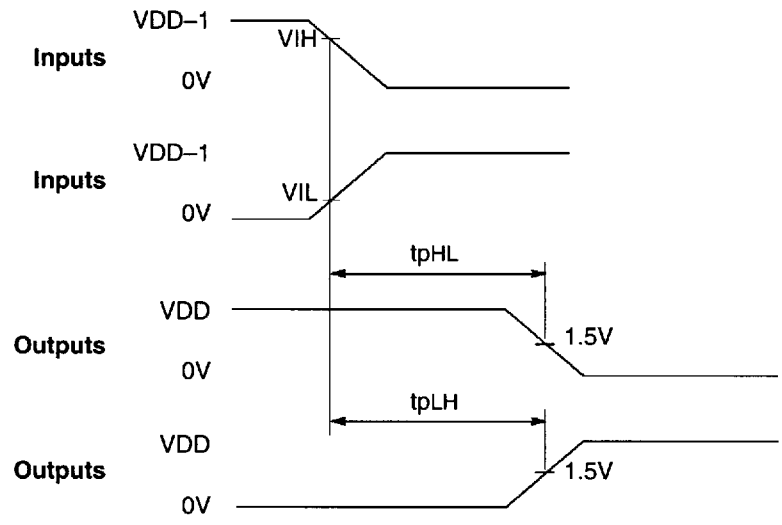


Figure 8.2 AC measurements timing waveforms

8.5 AC timing characteristics

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

Notes

1 Non-link pins; see section on links.

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

Table 8.4 Input and output edges

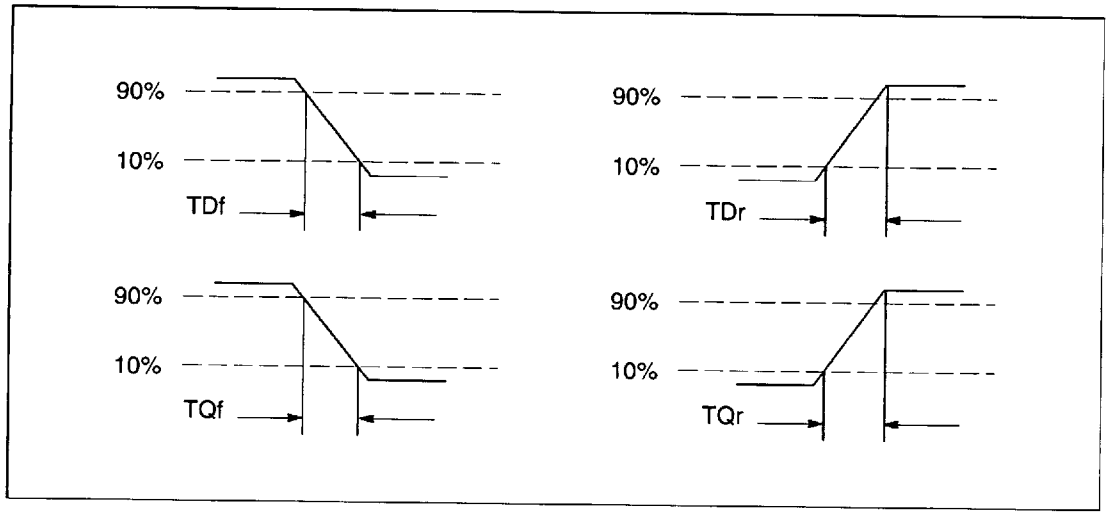


Figure 8.3 IMS T225 input and output edge timing

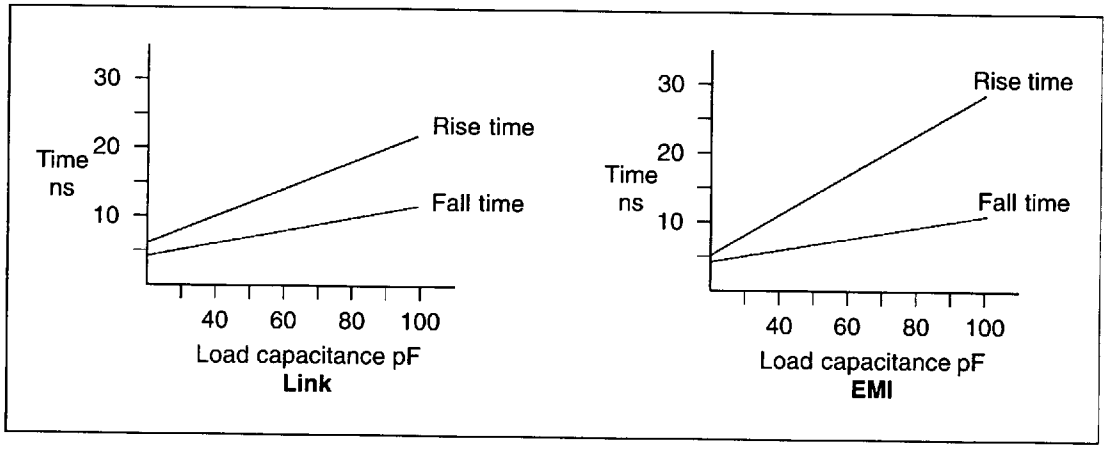


Figure 8.4 Typical rise/fall times

8.6 Power rating

Internal power dissipation P_{INT} of transputer and peripheral chips depends on **VDD**, as shown in figure 8.5. P_{INT} is substantially independent of temperature.

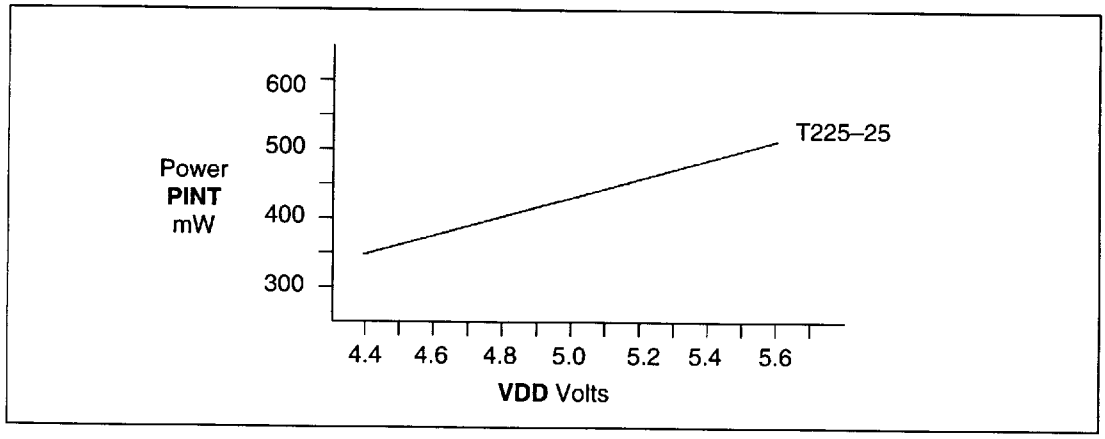


Figure 8.5 IMS T225 internal power dissipation vs VDD

Total power dissipation P_D of the chip is

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

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

Internal working temperature T_J of the chip is

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

where T_A is the external ambient temperature in °C and θ_{JA} is the junction-to-ambient thermal resistance in °C/W.

Further information about device thermal characteristics can be found in section 9.4.

9 Package details

9.1 68 pin grid array package

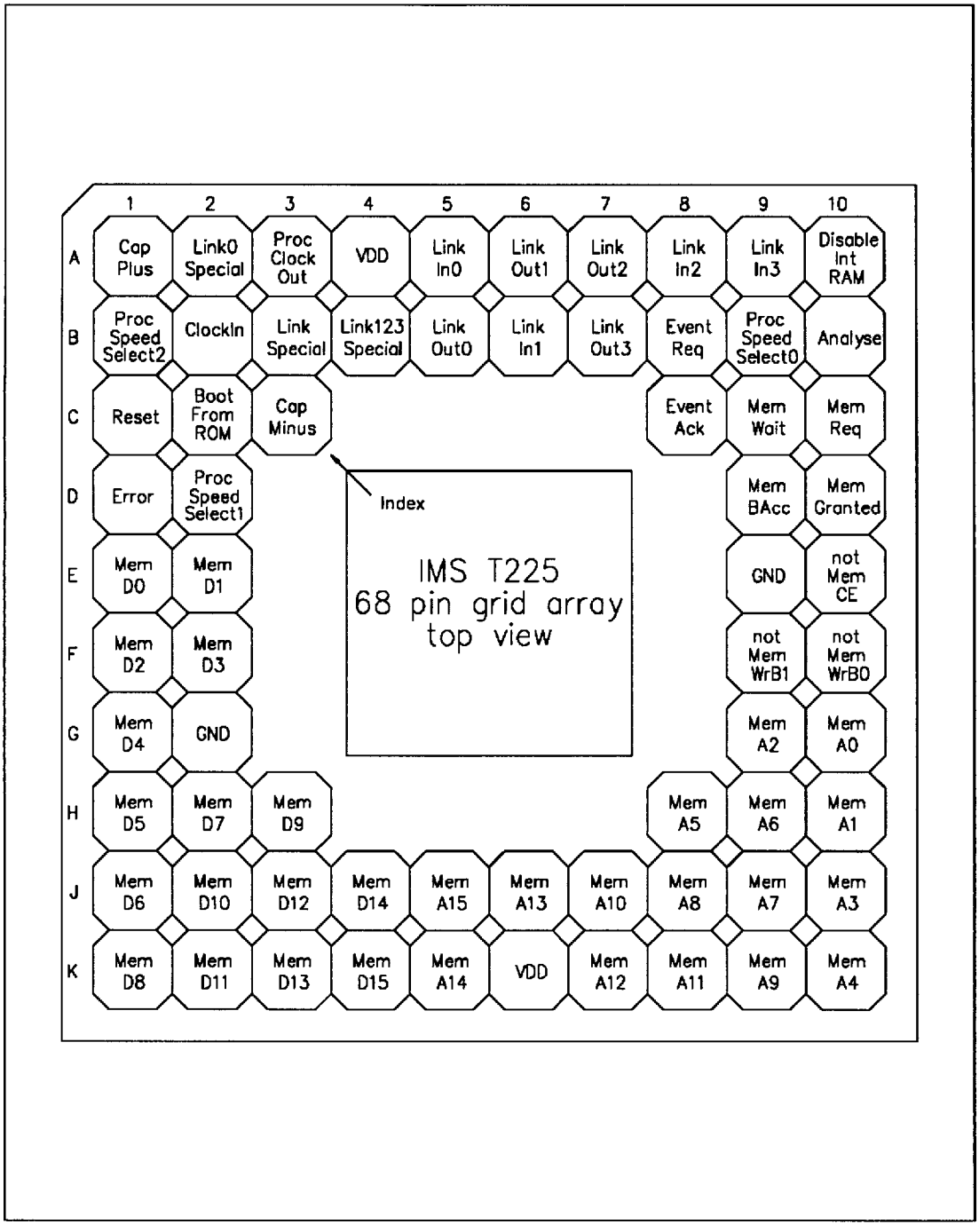


Figure 9.1 IMS T225 68 pin grid array package pinout

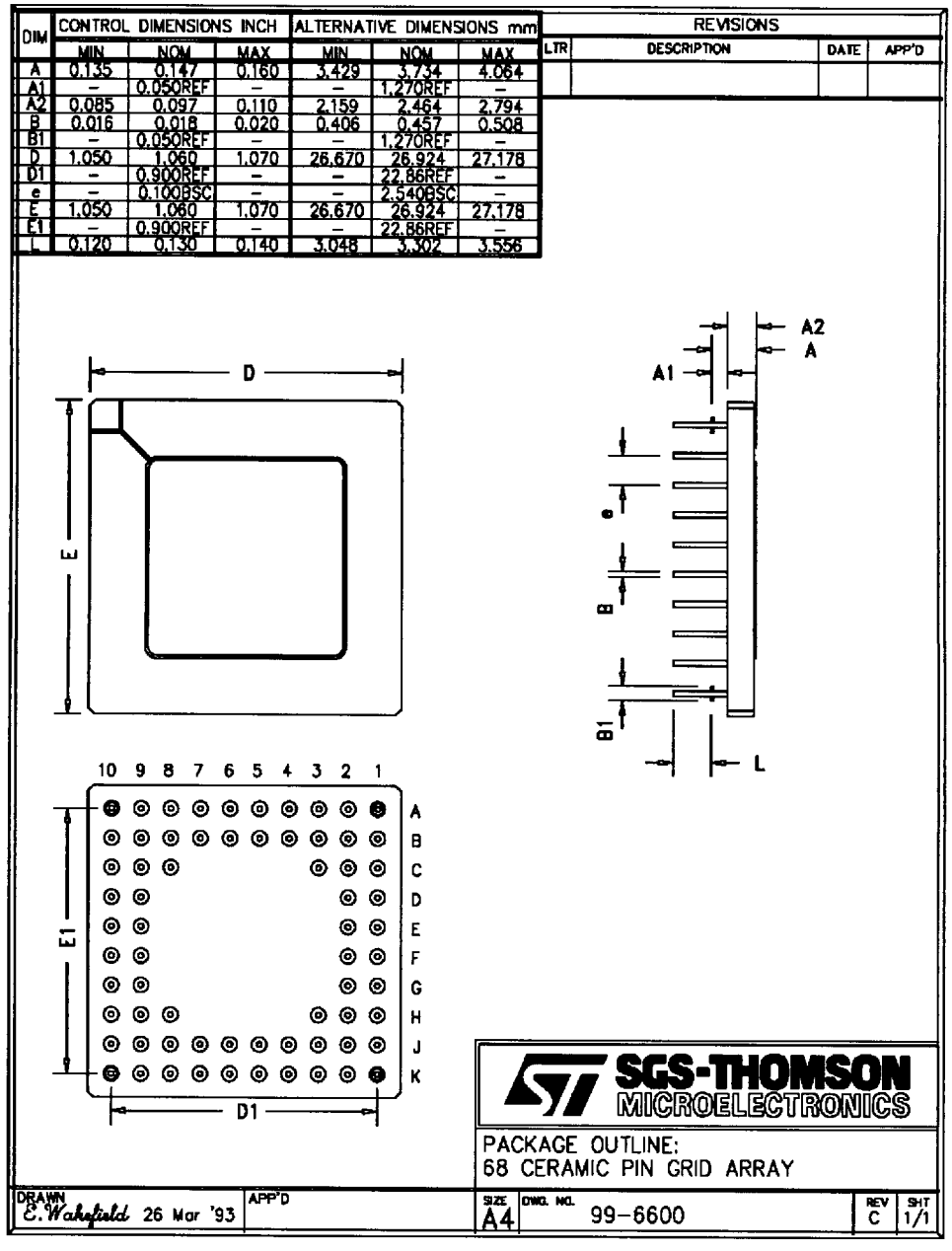


Figure 9.2 IMS T225 68 pin grid array package dimensions

9.2 68 pin PLCC J-bend package

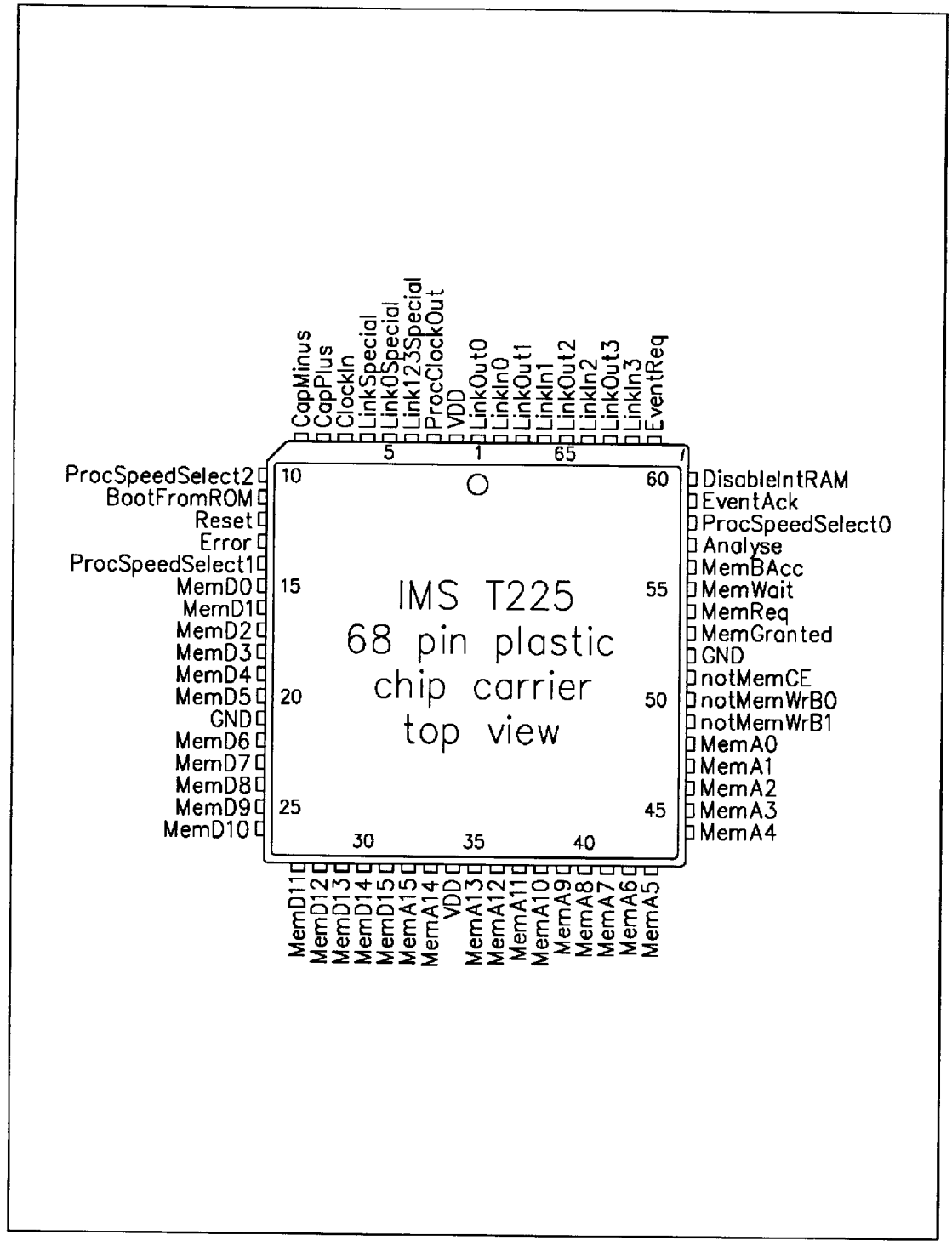
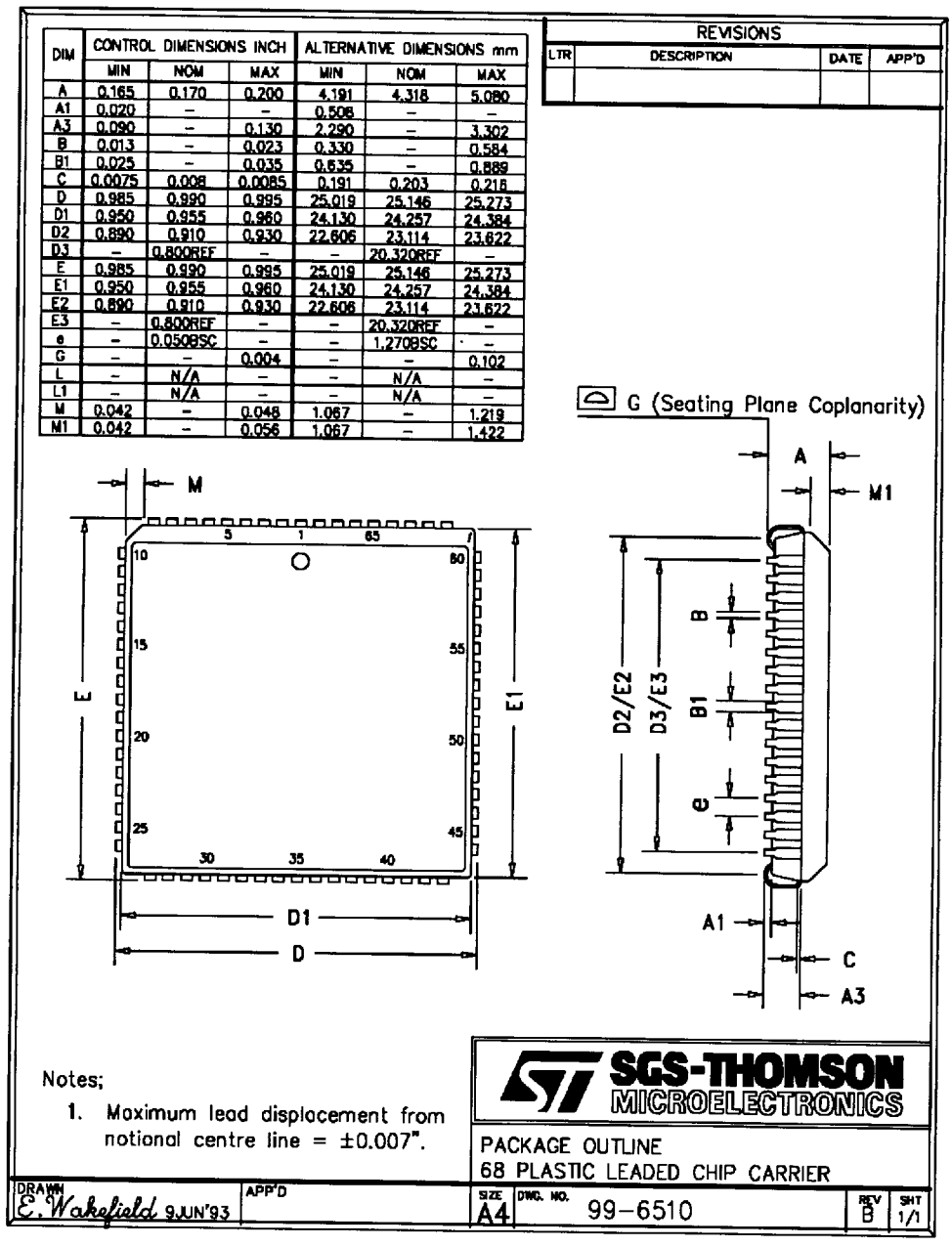


Figure 9.3 IMS T225 68 pin PLCC J-bend package pinout



DIM	CONTROL DIMENSIONS INCH			ALTERNATIVE DIMENSIONS mm		
	MIN	NOM	MAX	MIN	NOM	MAX
A	0.165	0.170	0.200	4.191	4.318	5.080
A1	0.020	–	–	0.508	–	–
A3	0.090	–	0.130	2.290	–	3.302
B	0.013	–	0.023	0.330	–	0.584
B1	0.025	–	0.035	0.635	–	0.889
C	0.0075	0.008	0.0085	0.191	0.203	0.216
D	0.985	0.990	0.995	25.019	25.146	25.273
D1	0.950	0.955	0.960	24.130	24.257	24.384
D2	0.890	0.910	0.930	22.606	23.114	23.622
D3	–	0.800REF	–	–	20.320REF	–
E	0.985	0.990	0.995	25.019	25.146	25.273
E1	0.950	0.955	0.960	24.130	24.257	24.384
E2	0.890	0.910	0.930	22.606	23.114	23.622
E3	–	0.800REF	–	–	20.320REF	–
e	–	0.050BSC	–	–	1.270BSC	–
G	–	–	0.004	–	–	0.102
L	–	N/A	–	–	N/A	–
L1	–	N/A	–	–	N/A	–
M	0.042	–	0.048	1.067	–	1.219
M1	0.042	–	0.056	1.067	–	1.422

Figure 9.4 IMS T225 68 pin PLCC J-bend package dimensions

9.3 100 pin cavity-up ceramic quad flat pack (CQFP) package

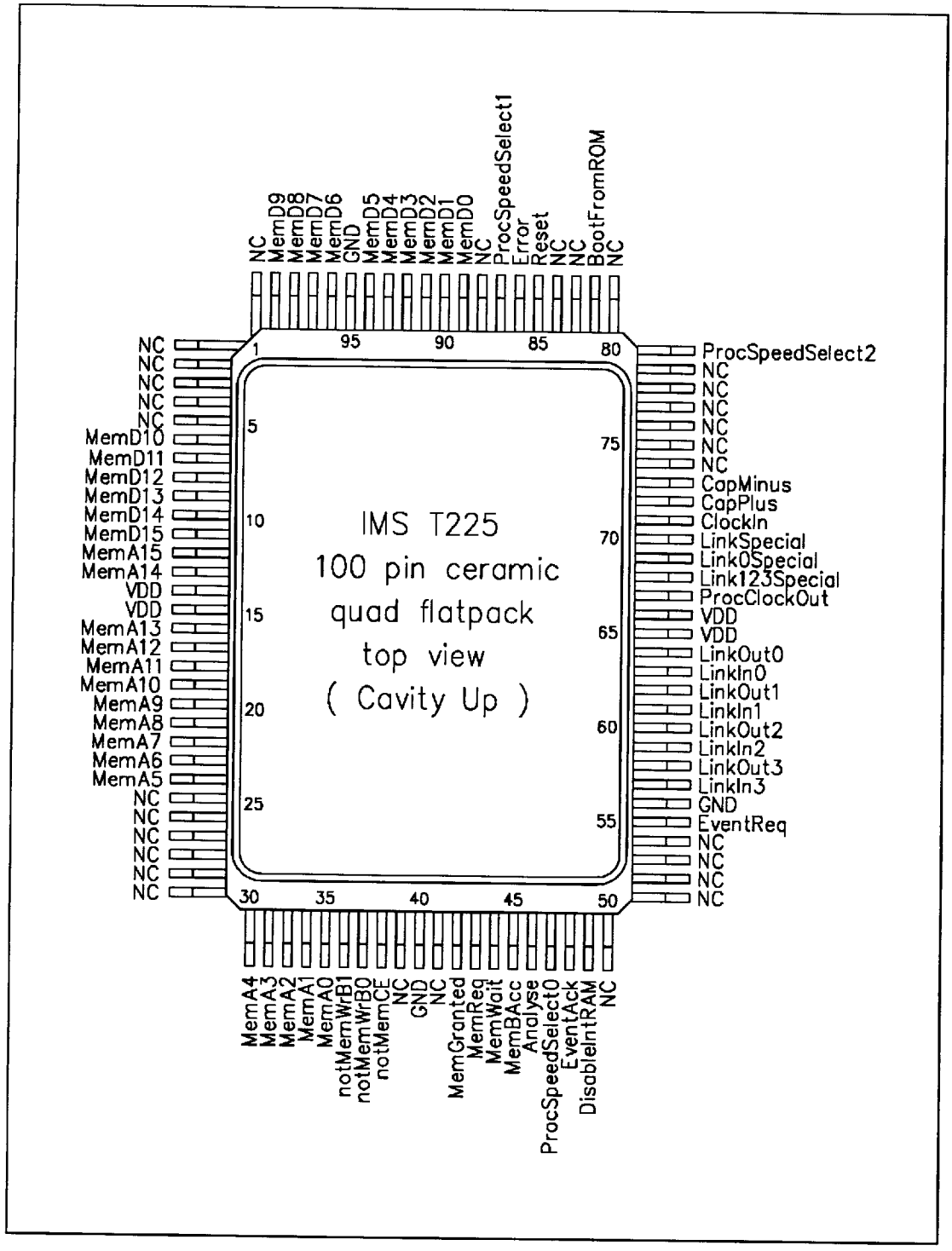


Figure 9.5 IMS T225 100 pin cavity-up ceramic quad flat pack package pinout

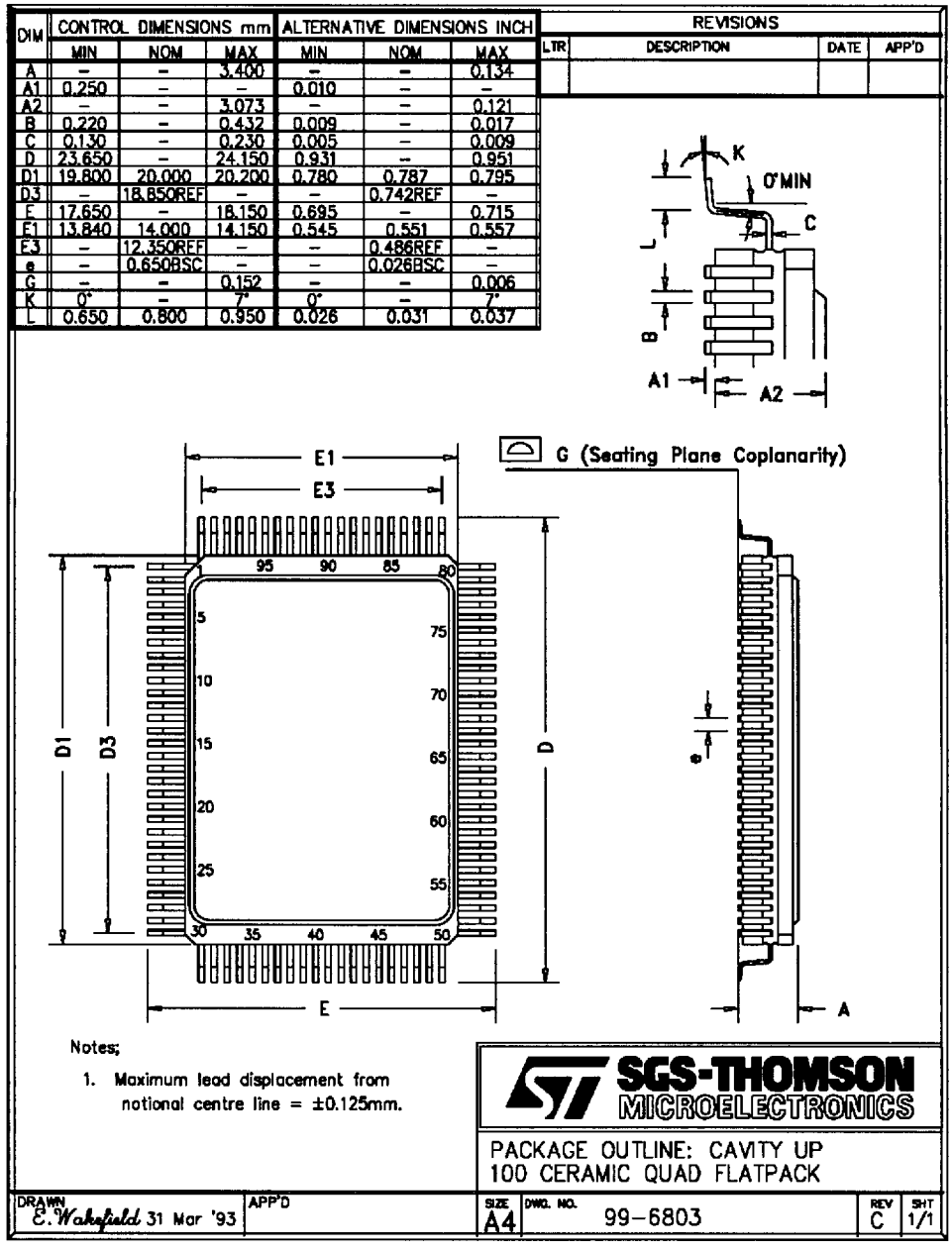


Figure 9.6 IMS T225 100 pin cavity-up ceramic quad flat pack package dimensions

9.4 Thermal specification

The IMS T225 is tested to a maximum silicon temperature of 100°C. For operation within the given specifications, the case temperature should not exceed 85°C.

For temperatures above 85°C the operation of the device cannot be guaranteed and reliability may be impaired.

For further information on reliability refer to the SGS–THOMSON Microelectronics Quality and Reliability Program.

10 Ordering

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

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

SGS-THOMSON designation	Processor clock speed	Processor cycle time	PLLx	Package
IMS T225-G25S	25.0 MHz	40ns	5.0	68 pin ceramic pin grid array
IMS T225-J25S	25.0 MHz	40ns	5.0	68 pin PLCC J–bend
IMS T225-F25S	25.0 MHz	40ns	5.0	100 pin ceramic quad flat pack

Table 10.1 IMS T225 ordering details

An extended temperature version is available, see the *IMS T225E Datasheet* for details.

11 Transputer instruction set summary

11.1 Introduction

The Function Codes table 11.9 (page 48) gives the basic function code set. Where the operand value is less than 16, a single byte encodes the complete instruction. If the operand value is greater than 15, one prefix instruction (*pfix*) is required for each additional four bits of the operand. If the operand is negative the first prefix instruction will be *nfix*. Examples of prefix coding are given in table 11.1.

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

Table 11.1 *prefix* coding

Tables 11.10 to 11.30 (pages 48–55) give details of the operation codes. Where an operation code is less than 16 (e.g. *add* operation code **05**), the operation can be stored as a single byte comprising the *operate* function code **F** and the operand (**5** in the example). Where an operation code is greater than 15 (e.g. *ladd* operation code **16**), the *prefix* function code **2** is used to extend the instruction.

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

Table 11.2 *operate* coding

11.1.1 Product identity numbers

The load device identity (*lddevid*) instruction (table 11.10) pushes the device type identity into the A register. Each product is allocated a unique group of numbers for use with the *lddevid* instruction. Product identity numbers are given in table 11.3.

Product	Identity numbers
IMS T425	0 to 9 inclusive
IMS T805	10 to 19 inclusive
IMS T225	40 to 49 inclusive
IMS T400	50 to 59 inclusive

Table 11.3 Product identity numbers

11.1.2 Floating point unit

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

In the Floating Point Operation Codes tables 11.23 to 11.29, a selector sequence code is indicated in the Memory Code column by **s**. The code given in the Operation Code column is the indirection code, the operand for the *ldc* instruction.

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

11.1.3 Notation

The Processor Cycles column refers to the number of periods **TPCLPCL** (refer to **ProcClockOut**) taken by an instruction executing in internal memory. The number of cycles is given for the basic operation only; where the memory code for an instruction is two bytes, the time for the *prefix* function (one cycle) should be added. Some instruction times vary. Where a letter is included in the cycles column it is interpreted from table 11.4.

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

Table 11.4 Instruction set interpretation

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

Ident	Feature	See section:
D	The instruction is a descheduling point	11.2
E	The instruction will affect the **Error** flag	11.3
F †	The instruction will affect the **FP_Error** flag	11.6
† applies to IMS T805 only		

Table 11.5 Instruction features

11.2 Descheduling points

The instructions in table 11.6 are the only ones at which a process may be descheduled. They are also the ones at which the processor will halt if the **Analyse** pin is asserted (refer to **Analyse** section).

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

Table 11.6 Descheduling point instructions

11.3 Error instructions

The instructions in table 11.7 are the only ones which can affect the *Error* flag directly. Note, however, that the floating point unit error flag *FP_Error* is set by certain floating point instructions (section 11.6), and that *Error* can be set from this flag by *fpcheckerror*.

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

Table 11.7 Error setting instructions

11.4 Debugging support

Table 11.20 (page 52) contains a number of instructions to facilitate the implementation of breakpoints. These instructions overload the operation of *j0*. Normally *j0* is a no-op which might cause descheduling. *Setj0break* enables the breakpointing facilities and causes *j0* to act as a breakpointing instruction. When breakpointing is enabled, *j0* swaps the current **Iptr** and **Wptr** with an **Iptr** and **Wptr** stored above MemStart. The *break* instruction does not cause descheduling, and preserves the state of the registers. It is possible to single step the processor at machine level using these instructions. Refer to *Support for debugging/breakpointing in transputers* (technical note 61) for more detailed information regarding debugger support.

11.5 Block move

The block move instructions (Table 11.21) move any number of bytes from any byte boundary in memory, to any other byte boundary, using the smallest possible number of word read, and word or part-word writes.

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

11.6 Floating point errors (IMS T805 only)

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

The instructions in table 11.8 are the only ones which can affect the floating point error flag *FP_Error*. *Error* is set from this flag by *fpcheckerror* if *FP_Error* is set.

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

Table 11.8 Floating point error setting instructions

11.7 General instructions

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

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

Table 11.9 Function codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
2A	22FA	testpranal	2	test processor analyzing	
3E	23FE	saveh	4	save high priority queue registers	
3D	23FD	savel	4	save low priority queue registers	
18	21F8	sthf	1	store high priority front pointer	
50	25F0	sthb	1	store high priority back pointer	
1C	21FC	stlf	1	store low priority front pointer	
17	21F7	stlb	1	store low priority back pointer	
54	25F4	sttimer	1	store timer	
17C	2127FC	lddevid	1	load device identity	
7E	27FE	ldmemstartval	1	load value of memstart address	

Table 11.10 Processor initialisation operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles		Name	DEF
			16-bit devices	32-bit devices		
46	24F6	and	1	1	and	
4B	24FB	or	1	1	or	
33	23F3	xor	1	1	exclusive or	
32	23F2	not	1	1	bitwise not	
41	24F1	shl	**n**+2	**n**+2	shift left	
40	24F0	shr	**n**+2	**n**+2	shift right	
05	F5	add	1	1	add	E
0C	FC	sub	1	1	subtract	E
53	25F3	mul	23	38	multiply	E
72 †	27F2	fmul		35	fractional multiply (no rounding)	E
				40	fractional multiply (rounding)	E
2C	22FC	div	24	39	divide	E
1F	21FF	rem	21	37	remainder	E
09	F9	gt	2	2	greater than	
04	F4	diff	1	1	difference	
52	25F2	sum	1	1	sum	
08	F8	prod	**b**+4	**b**+4	product for positive register **A**	
08	F8	prod	**m**+5	**m**+5	product for negative register **A**	
† does not apply to IMS T225						

Table 11.11 Arithmetic/logical operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles		Name	DEF
			16-bit devices	32-bit devices		
16	21F6	ladd	2	2	long add	E
38	23F8	lsub	2	2	long subtract	E
37	23F7	lsum	3	3	long sum	
4F	24FF	ldiff	3	3	long diff	
31	23F1	lmul	17	33	long multiply	
1A	21FA	ldiv	19	35	long divide	E
36	23F6	lshl	**n**+3	**n**+3	long shift left (**n**<32)	† (**n**<16)
			n–12	**n**–28	long shift left(**n**≥32)	† (**n**≥16)
35	23F5	lshr	**n**+3	**n**+3	long shift right (**n**<32)	† (**n**<16)
			n–12	**n**–28	long shift right (**n**≥32)	† (**n**≥16)
19	21F9	norm	**n**+5	**n**+5	normalise (**n**<32)	† (**n**<16)
			n–10	**n**–26	normalise (**n**≥32)	† (**n**≥16)
			3	3	normalise (**n**=64)	† (**n**=32)
† for IMS T225						

Table 11.12 Long arithmetic operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
00	F0	rev	1	reverse	
3A	23FA	xword	4	extend to word	
56	25F6	cword	5	check word	E
1D	21FD	xdble	2	extend to double	
4C	24FC	csngl	3	check single	E
42	24F2	mint	1	minimum integer	
5A	25FA	dup	1	duplicate top of stack	
79	27F9	pop	1	pop processor stack	

Table 11.13 General operation codes

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

Table 11.14 Indexing/array operation codes

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

Table 11.15 Timer handling operation codes

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

Table 11.16 Input/output operation codes

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

Table 11.17 Control operation codes

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

Table 11.18 Scheduling operation codes

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

Table 11.19 Error handling operation codes

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

Table 11.20 Debugger support codes

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

Table 11.21 2D block move operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
74	27F4	crcword	35	calculate crc on word	
75	27F5	crcbyte	11	calculate crc on byte	
76	27F6	bitcnt	b+2	count bits set in word	
77	27F7	bitrevword	36	reverse bits in word	
78	27F8	bitrevnbits	n+4	reverse bottom n bits in word	

Table 11.22 CRC and bit operation codes

11.8 Floating point instructions

11.9 Floating point instructions for IMS T805 only

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

Table 11.23 Floating point load/store operation codes

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

Table 11.24 Floating point general operation codes

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

Table 11.25 Floating point rounding operation codes

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

Table 11.26 Floating point error operation codes

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

Table 11.27 Floating point comparison operation codes

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

Table 11.28 Floating point conversion operation codes

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

Table 11.29 Floating point arithmetic operation codes

11.10 Floating point instructions for IMS T400 and IMS T425 only

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

Table 11.30 Floating point support operation codes