SGS-THOMSON MICROELECTRONICS

IMS T805

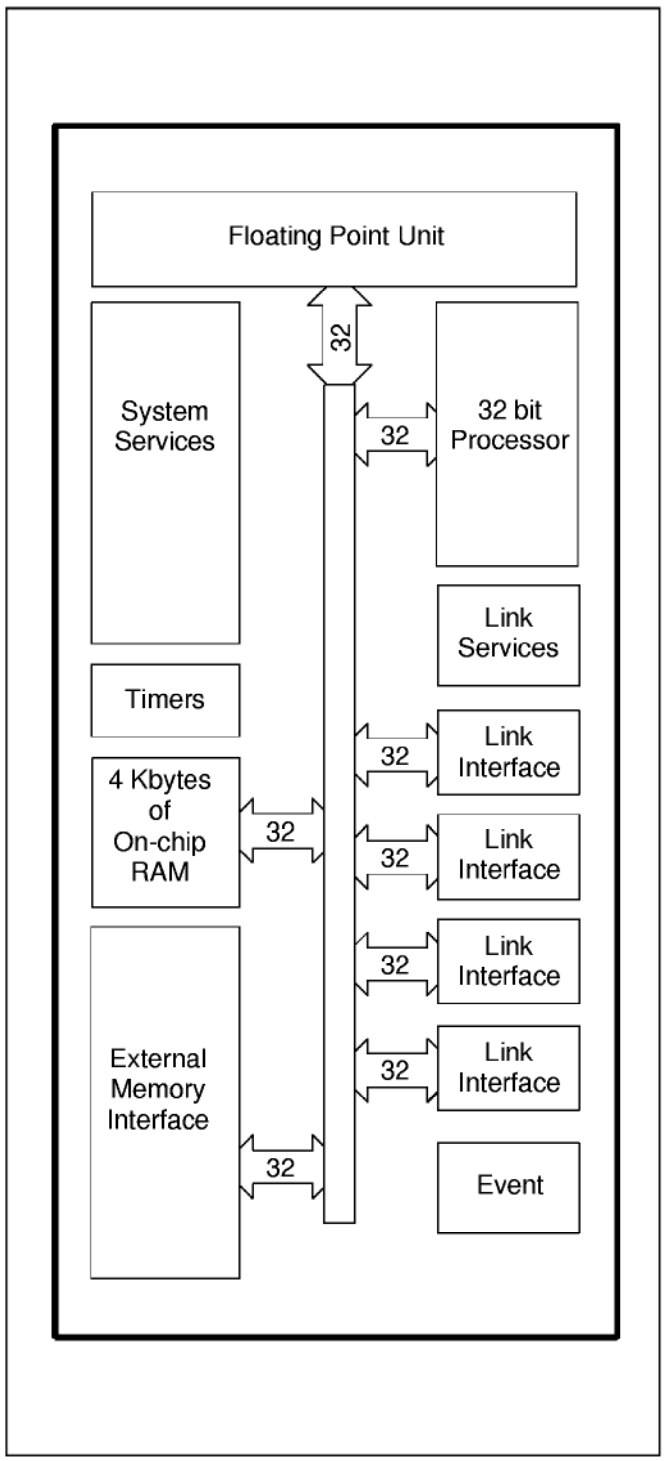
32-bit floating-point transputer

FEATURES

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

APPLICATIONS

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



February 1997 42 1440 07

Contents

1 Introduction

The IMS T805 transputer is a 32 bit CMOS microcomputer with a 64 bit floating point unit and graphics support. It has 4 Kbytes on-chip RAM for high speed processing, a configurable memory interface and four standard INMOS communication links. The instruction set achieves efficient implementation of high level languages 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 T805 operation is split into the basic blocks shown in figure 1.1.

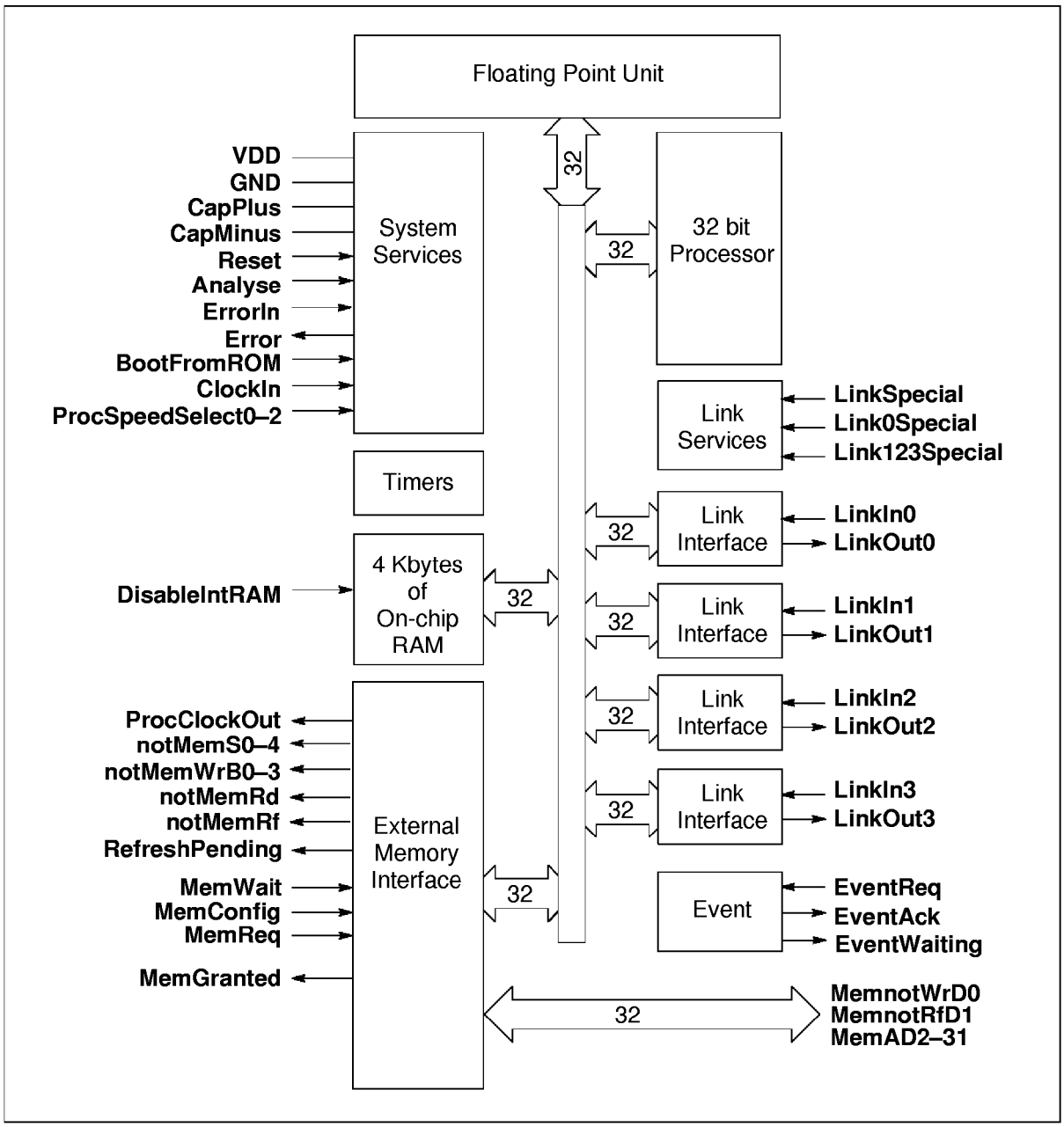


Figure 1.1 IMS T805 block diagram

The processor speed of a device can be pin-selected in stages from 20 MHz up to the maximum allowed for the part. A device running at 25 MHz achieves an instruction throughput of 25 MIPS peak and 12 MIPS sustained.

The IMS T805 provides high performance arithmetic and floating point operations. The 64 bit floating point unit provides single and double length operation to the ANSI-IEEE 754-1985 standard for floating point

arithmetic. It is able to perform floating point operations concurrently with the processor, sustaining a rate of 2.8 Mflops at a processor speed of 25 MHz.

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

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

The IMS T805 can directly access a linear address space of 4 Gbytes. The 32 bit wide memory interface uses multiplexed data and address lines and provides a data rate of up to 4 bytes every 120 nanoseconds 33 Mbytes/sec) for a 25 MHz device. A configurable memory controller provides all timing, control and DRAM refresh signals for a wide variety of mixed memory systems.

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

The IMS T805 is pin compatible with the IMS T800, as the extra inputs used are all held to ground on the IMS T800.

The 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 12.

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

Figure 1.2 shows datapaths for the IMS T805.

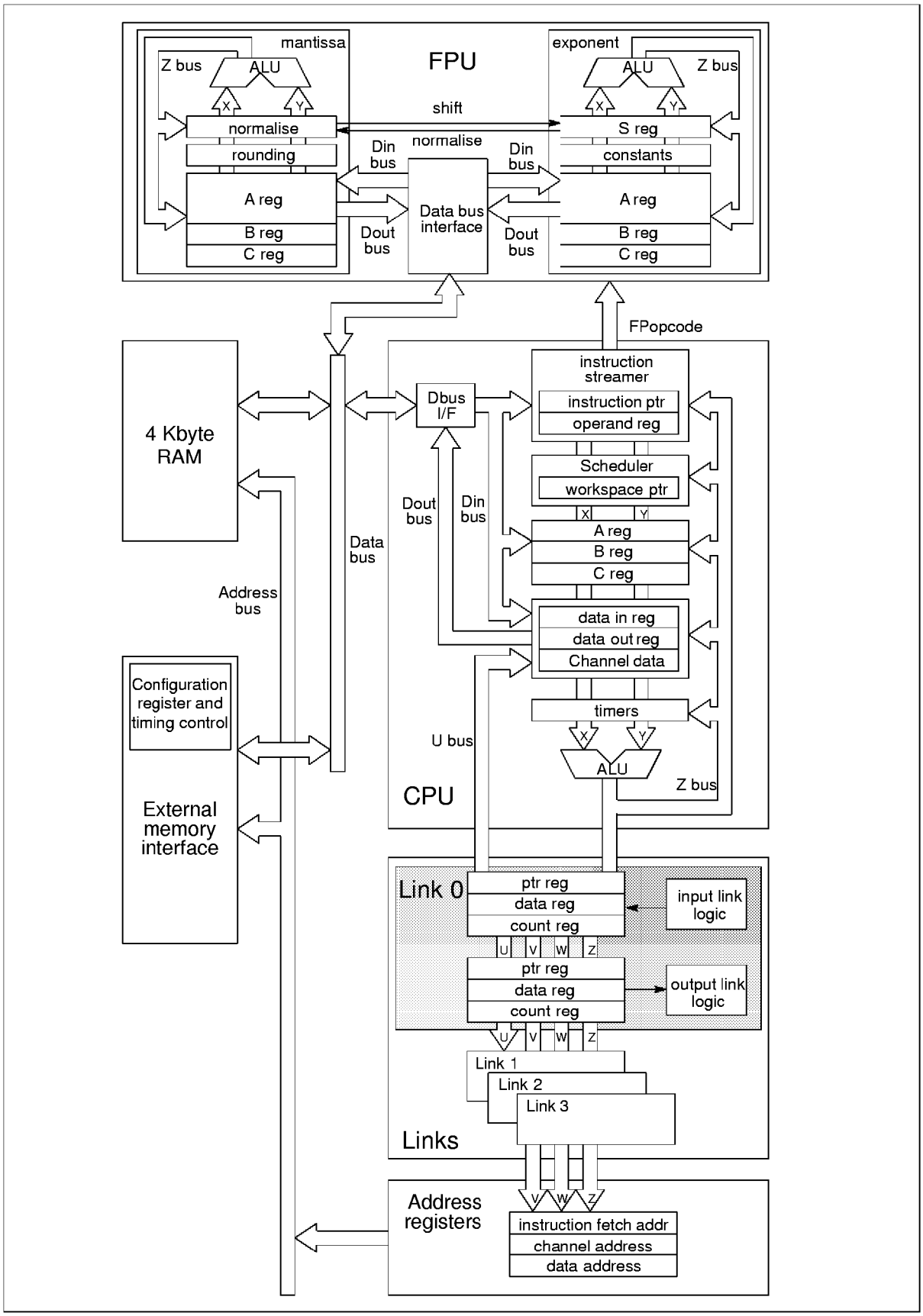


Figure 1.2 IMS T805 internal datapaths

2 Pin designations

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

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

Table 2.1 IMS T805 system services

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

Table 2.2 IMS T805 external memory interface

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

Table 2.3 IMS T805 event

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

Table 2.4 IMS T805 link

3 Floating point unit

The 64 bit FPU provides single and double length arithmetic to floating point standard ANSI-IEEE 754-1985. It is able to perform floating point arithmetic concurrently with the central processor unit (CPU), sustaining 2.8 Mflops on a 25 MHz device. All data communication between memory and the FPU occurs under control of the CPU.

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

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

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

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

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

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

fpuabs; *ldc* *fpentry;*

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

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

Table 3.1 *fpentry* coding

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

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

The FPU has its own error flag *FP_Error*. This reflects the state of evaluation within the FPU and is set in circumstances where invalid operations, division by zero or overflow exceptions to the ANSI-IEEE 754-1985 standard would be flagged (page 62). *FP_Error* is also set if an input to a floating point operation is infinite or is not a number (NaN). The *FP_Error* flag can be set, tested and cleared without affecting the main *Error* flag, but can also set *Error* when required (page 61). Depending on how a program is compiled, it is possible for both unchecked and fully checked floating point arithmetic to be performed.

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

Operation	T805-25	
	Single length	**Double length**
add	280 ns	280 ns
subtract	280 ns	280 ns
multiply	440 ns	800 ns
divide	680 ns	1280 ns
Timing is for operations where both operands are normalised fp numbers.		

Table 3.2 Typical floating point operation times for IMS T805

4 System services

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

4.1 Power

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

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

4.2 CapPlus, CapMinus

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

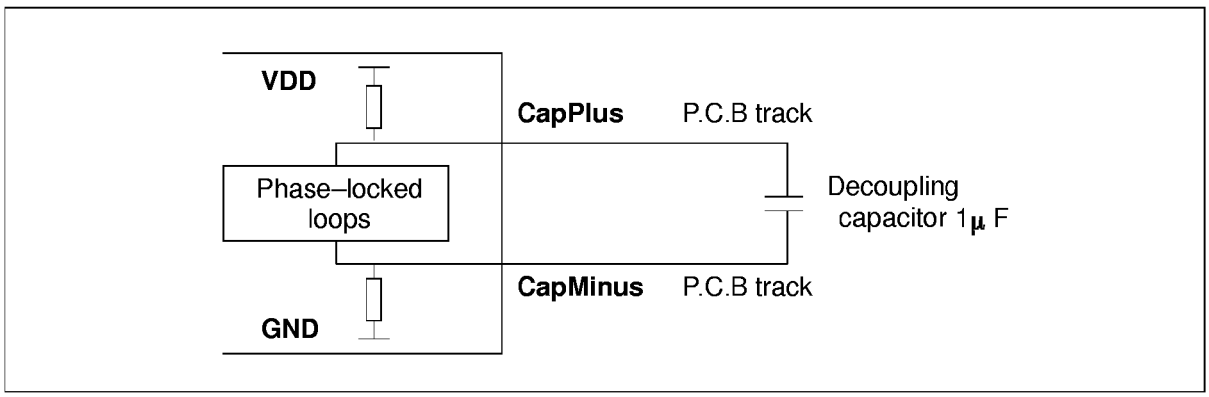


Figure 4.1 Recommended PLL decoupling

4.3 ClockIn

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

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

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

Symbol	Parameter	T805-25			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 9.3).

Table 4.1 Input clock

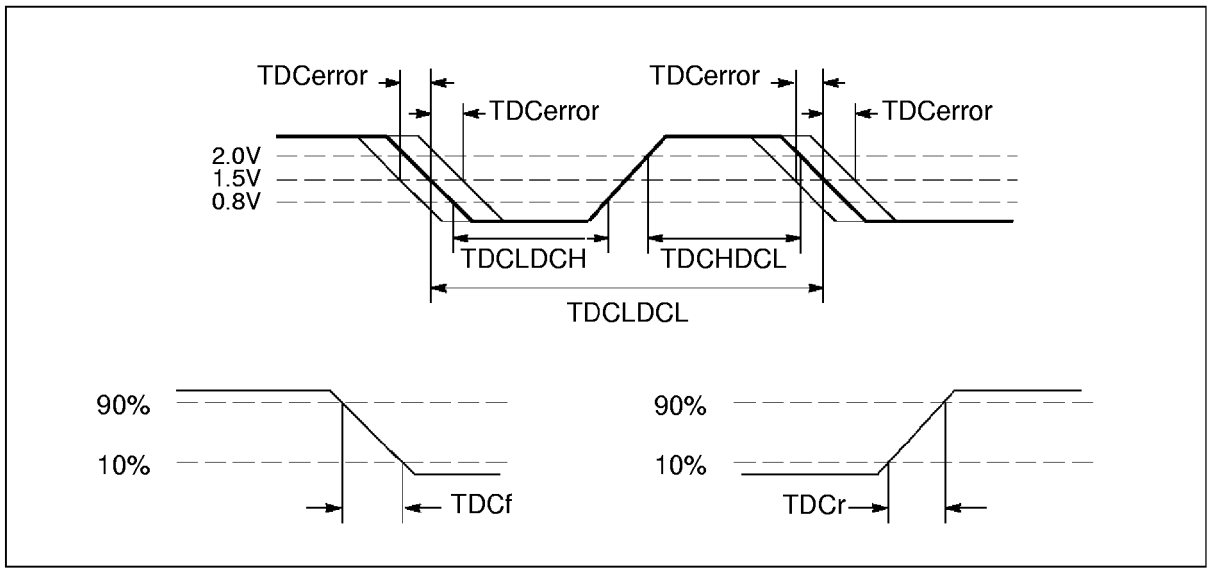


Figure 4.2 ClockIn timing

4.4 ProcSpeedSelect0–2

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

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

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

Table 4.2 Processor speed selection

4.5 Bootstrap

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

If **BootFromROM** is connected high (e.g. to **VDD**) the transputer starts to execute code from the top two bytes in external memory, at address #7FFFFFFE. This location should contain a backward jump to a program in ROM. Following this access, **BootFromROM** may be taken low if required. The processor is in the low priority state, and the *W* register points to *MemStart* (page 15).

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

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

4.6 Peek and poke

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

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

4.7 Reset

Reset can go high with **VDD**, but must at no time exceed the maximum specified voltage for **VIH**. After **VDD** is valid **ClockIn** should be running for a minimum period **TDCVRL** before the end of **Reset**. The falling edge of **Reset** initialises the transputer, triggers the memory configuration sequence and starts the bootstrap routine. Link outputs are forced low during reset; link inputs and **EventReq** should be held low. Memory request (DMA) must not occur whilst **Reset** is high but can occur before bootstrap (page 32). After the end of **Reset** there will be a delay of 144 periods of **ClockIn** (figure 4.3). Following this, the **MemWrD0**, **MemRfD1** and **MemAD2-31** pins will be scanned to check for the existence of a pre-programmed memory interface configuration (page 35). This lasts for a further 144 periods of **ClockIn**. Regardless of whether a configuration was found, 36 configuration read cycles will then be performed on external memory using the default memory configuration (page 37), in an attempt to access the external configuration ROM. A delay will then occur, its period depending on the actual configuration. Finally eight complete and consecutive refresh cycles will initialise any dynamic RAM, using the new memory configuration. If the memory configuration does not enable refresh of dynamic RAM the refresh cycles will be replaced by an equivalent delay with no external memory activity.

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

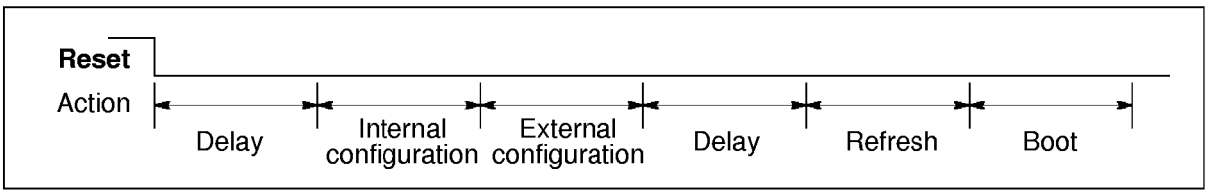


Figure 4.3 IMS T805 post-reset sequence

4.8 Analyse

If **Analyse** is taken high when the transputer is running, the transputer will halt at the next descheduling point (page 61). From **Analyse** being asserted, the processor will halt within three time slice periods plus the time taken for any high priority process to complete. As much of the transputer status is maintained as is necessary to permit analysis of the halted machine. Processor flags *Error*, *HaltOnError* and *EnableJ0Break* are normally cleared at reset on the IMS T805; however, if **Analyse** is asserted the flags are not altered. Memory refresh continues.

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

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

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

Table 4.3 Register values after Analyse

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

Notes

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

2 At power-on reset.

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

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

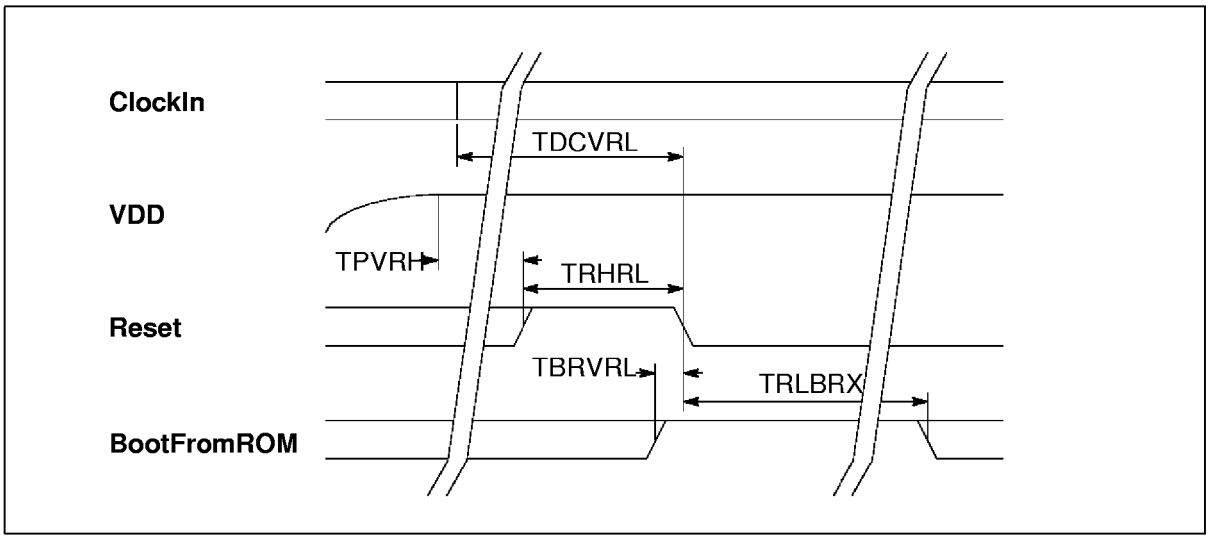


Figure 4.4 Transputer **Reset** timing with **Analyse** low

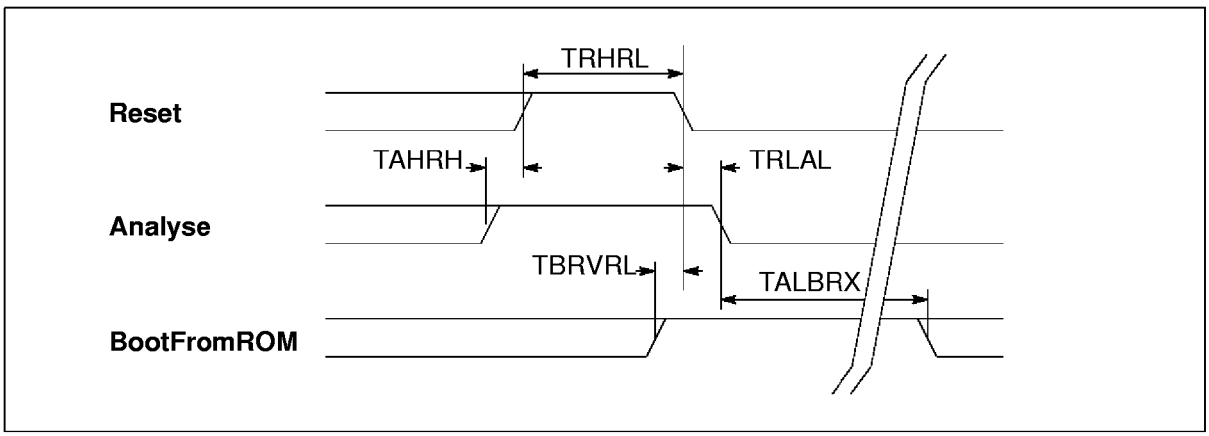


Figure 4.5 Transputer **Reset**, **Analyse** and **BootFromROM** timing

4.9 Error, ErrorIn

The **Error** pin carries the OR'ed output of the internal *Error* flag and the **ErrorIn** input. If **Error** is high it indicates either that **ErrorIn** is high or that an error was detected in one of the processes. An internal error can be caused, for example, by arithmetic overflow, divide by zero, array bounds violation or software setting the flag directly (page 61). It can also be set from the floating point unit under certain circumstances (pages 7, 62). 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 12). A process can be programmed to stop if the *Error* flag is set; it cannot then transmit erroneous data to other processes, but processes which do not require that data can still be scheduled. Eventually all processes which rely, directly or indirectly, on data from the process in error will stop through lack of data. **ErrorIn** does not directly affect the status of a processor in any way.

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

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

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

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

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

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

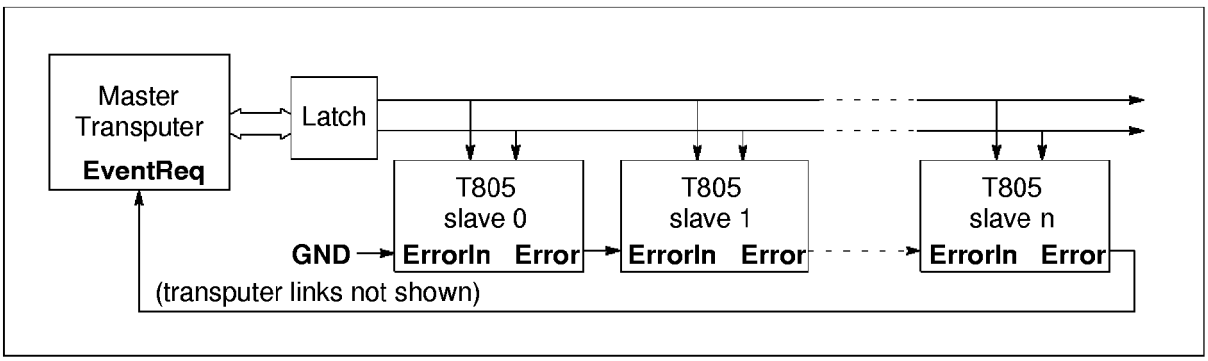


Figure 4.6 Error handling in a multi-transputer system

5 Memory

The IMS T805 has 4 Kbytes of fast internal static memory for high rates of data throughput. Each internal memory access takes one processor cycle **ProcClockOut** (page 19). The transputer can also access 4 Gbytes of external memory space. Internal and external memory are part of the same linear address space. Internal RAM can be disabled by holding **DisableIntRAM** high. All internal addresses are then mapped to external RAM. This pin should not be altered after **Reset** has been taken low.

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

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

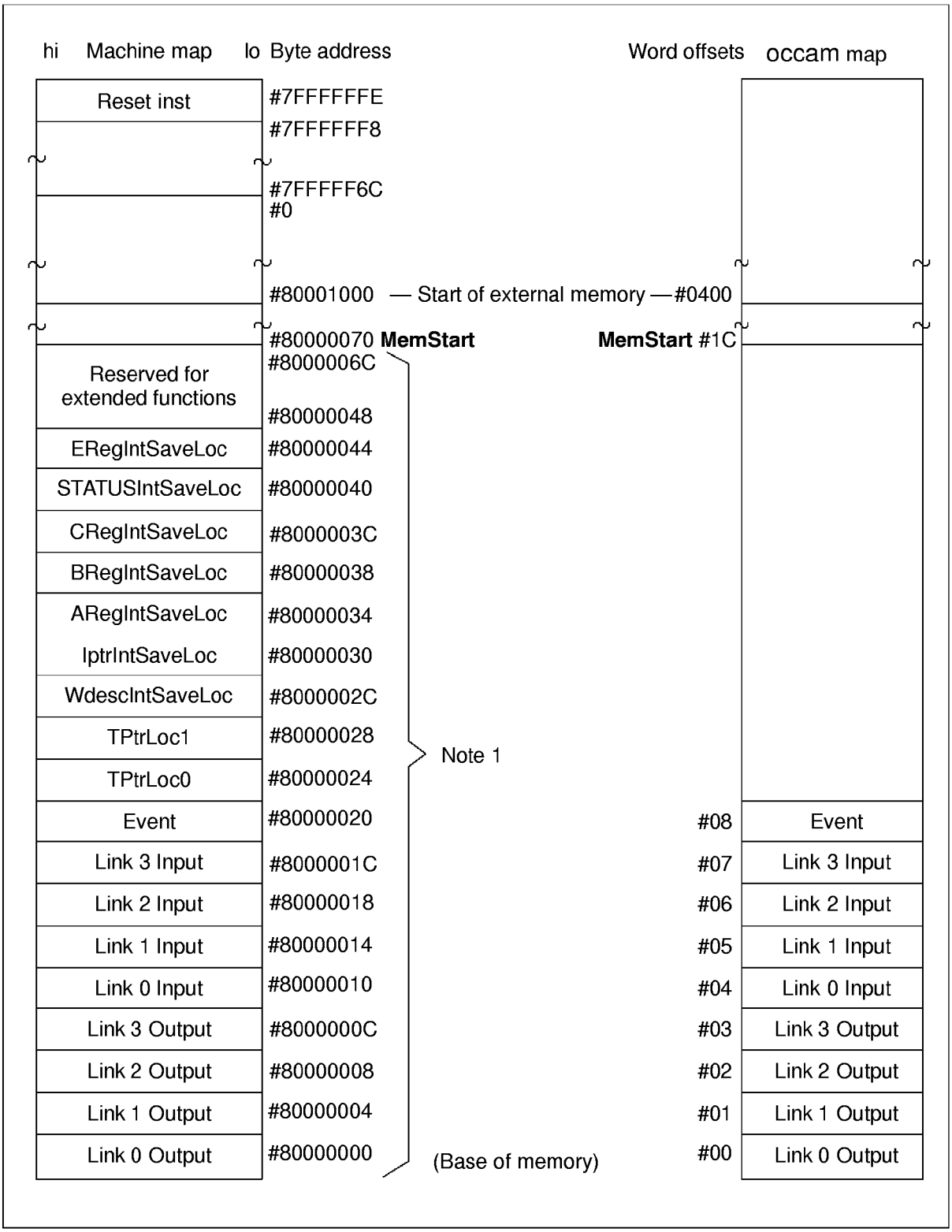
Internal memory starts at the most negative address #80000000 and extends to #80000FFF. User memory begins at #80000070; this location is given the name *MemStart*. An instruction *ldmemstartval* is provided to obtain the value of **MemStart**.

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

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

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

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



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 12). For details see *Transputer Instruction Set – A Compiler Writers' Guide*.

Figure 5.1 IMS T805 memory map

6 External memory interface

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

There are 17 internal configurations which can be selected by a single pin connection (page 35). If none are suitable the user can configure the interface to specific requirements, as shown on page 37.

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

T1 Address setup time before address valid strobe.

T2 Address hold time after address valid strobe.

T3 Read cycle tristate or write cycle data setup.

T4 Extendable data setup time.

T5 Read or write data.

T6 Data hold.

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

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

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

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

The bus activity is not adequate to trace the internal operation of the transputer in full, but may be used for hardware debugging in conjunction with peek and poke (page 11).

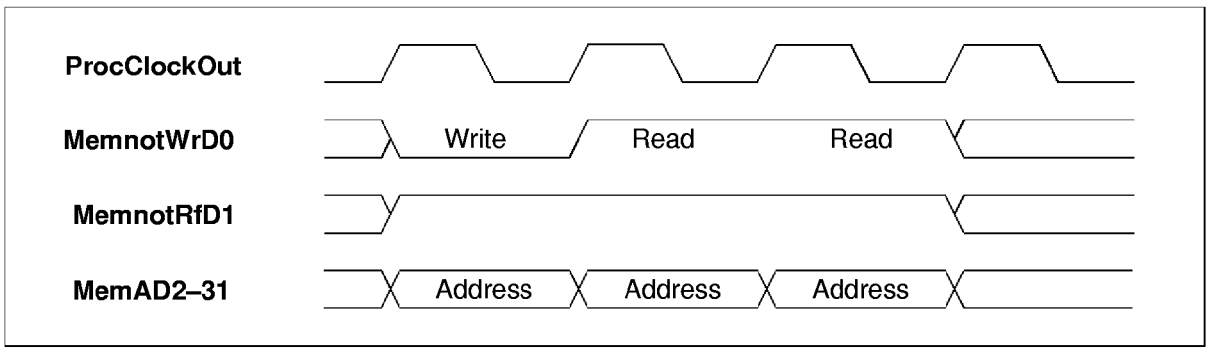


Figure 6.1 IMS T805 bus activity for internal memory cycle

6.1 Pin functions

6.1.1 MemAD2–31

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

6.1.2 notMemRd

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

6.1.3 MemnotWrD0

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

6.1.4 notMemWrB0–3

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

6.1.5 notMemS0–4

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

6.1.6 MemWait

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

6.1.7 MemnotRfD1

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

6.1.8 notMemRf

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

6.1.9 RefreshPending

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

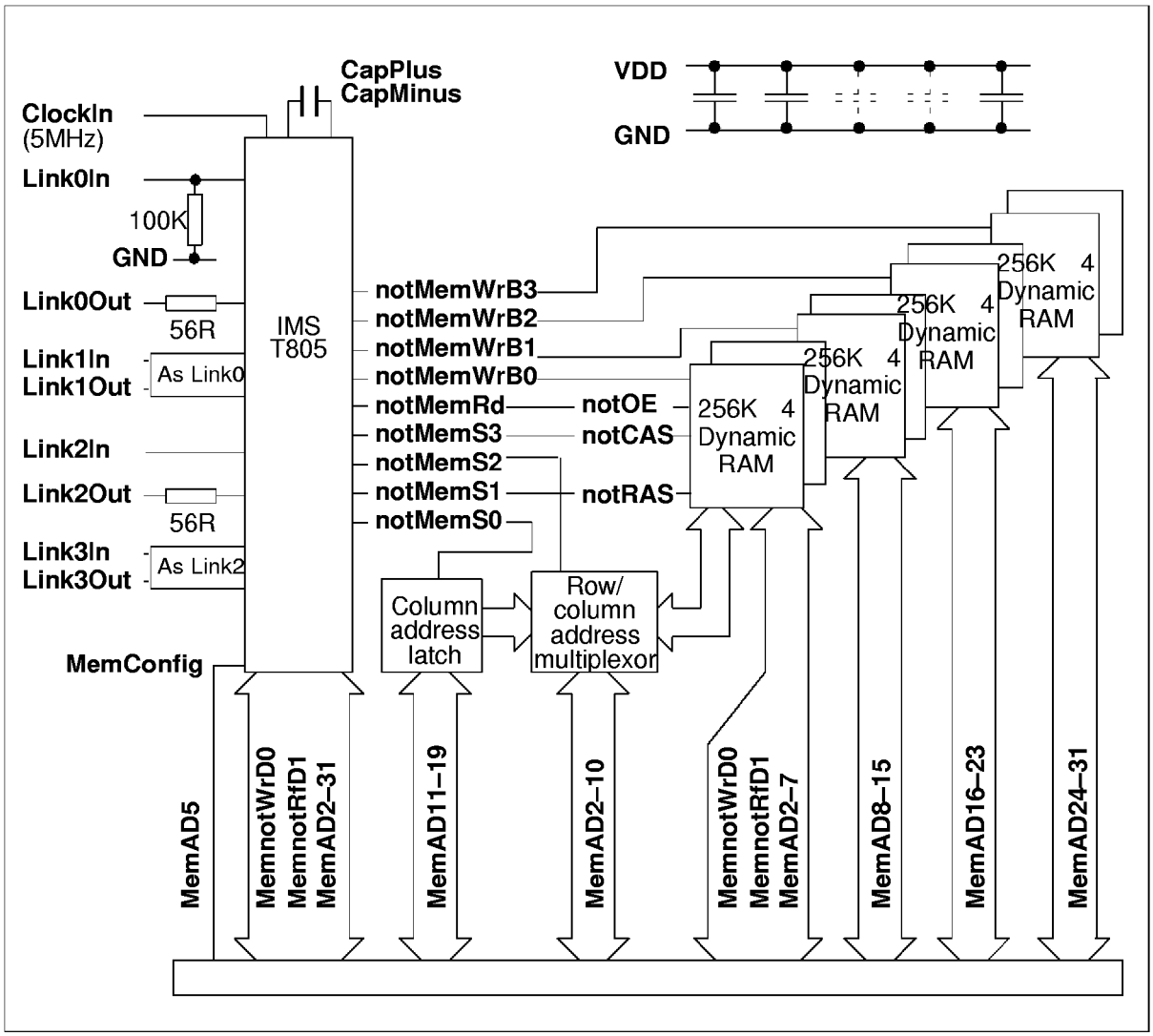


Figure 6.2 IMS T805 application

6.1.10 MemReq, MemGranted

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

6.1.11 MemConfig

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

6.1.12 ProcClockOut

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

TPCLPCL = TDCLDCL / PLLx

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

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

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

Symbol	Parameter	T805-25		Units	Notes
		Min	Max		
TPCLPCL	**ProcClockOut** period	38	42	ns	
TPCHPCL	**ProcClockOut** pulse width high	8.5	23.5	ns	
TPCLPCH	**ProcClockOut** pulse width low	**a**		ns	2,3
Tm	**ProcClockOut** half cycle	19	21	ns	
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 6.1 **ProcClockOut**

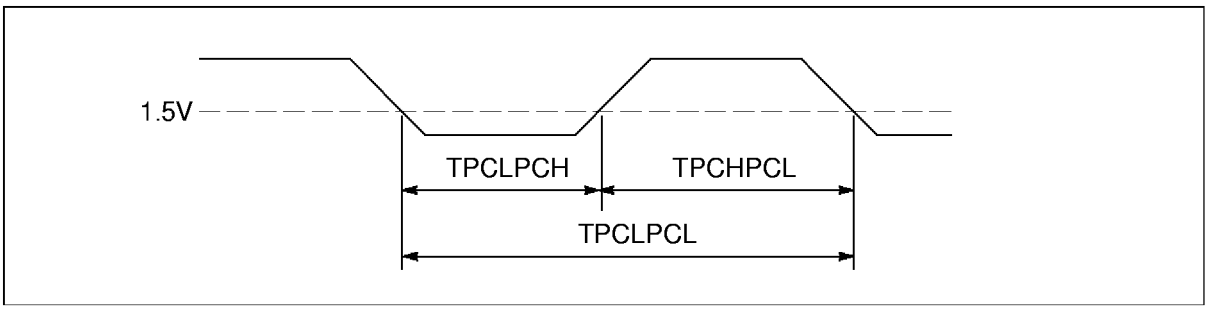


Figure 6.3 IMS T805 **ProcClockOut** timing

6.2 Read cycle

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

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

The leading edge of **notMemS1** is always coincident with the start of **T2**, but its duration may be configured to be from zero to 31 periods **Tm**. Regardless of the configured duration, the strobe will terminate no later than the end of **T6**. The strobe is sometimes programmed to extend beyond the normal end of **Tmx**. When wait states are inserted into an EMI cycle the end of **Tmx** is delayed, but the potential active duration of the strobe is not altered. Thus the strobe can be configured to terminate relatively early under certain conditions (page 27). If **notMemS1** is configured to be zero it will never go low.

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

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

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

Notes

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

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

Table 6.2 Read

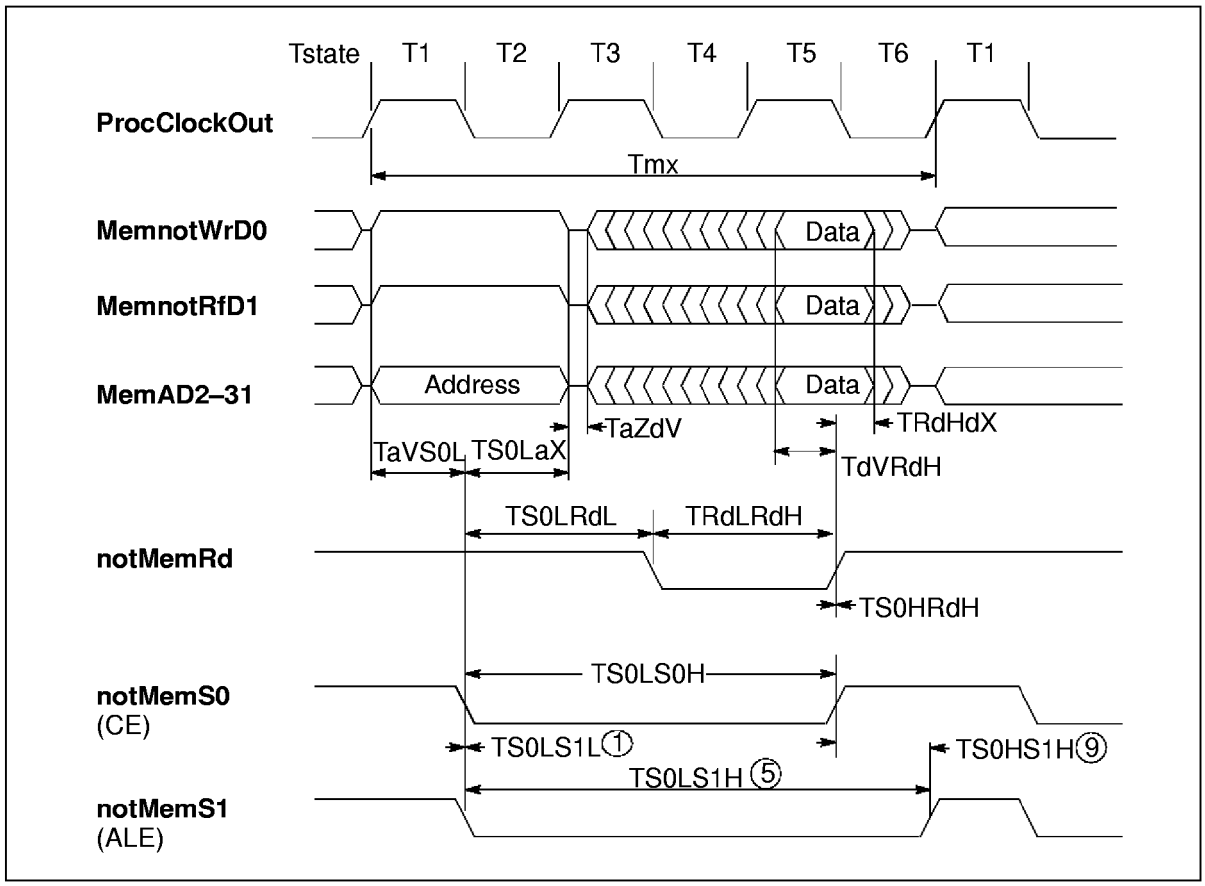


Figure 6.4 IMS T805 external read cycle: static memory

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

Notes

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

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

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

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

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

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

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

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

Table 6.3 IMS T805 strobe timing

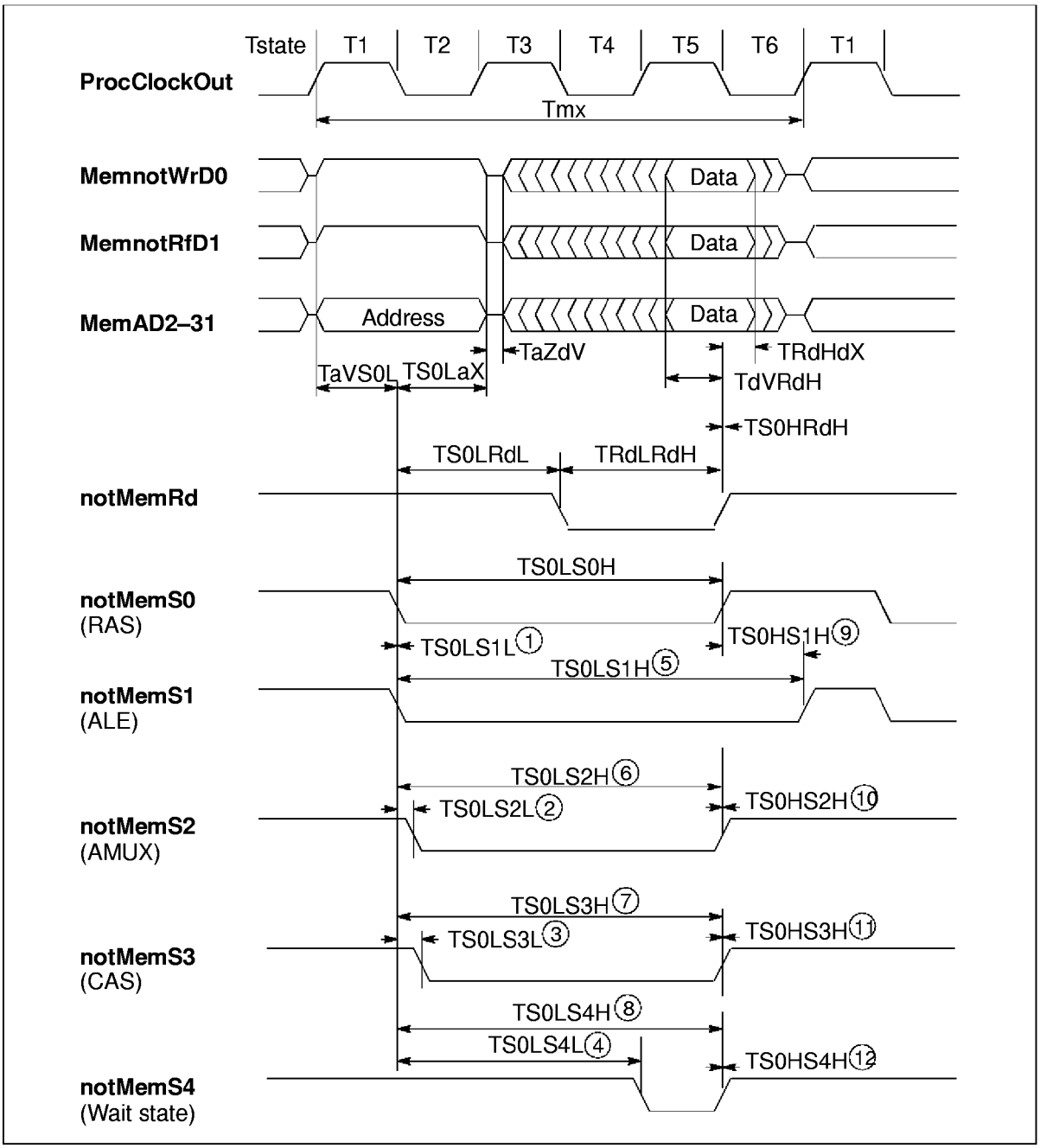


Figure 6.5 IMS T805 external read cycle: dynamic memory

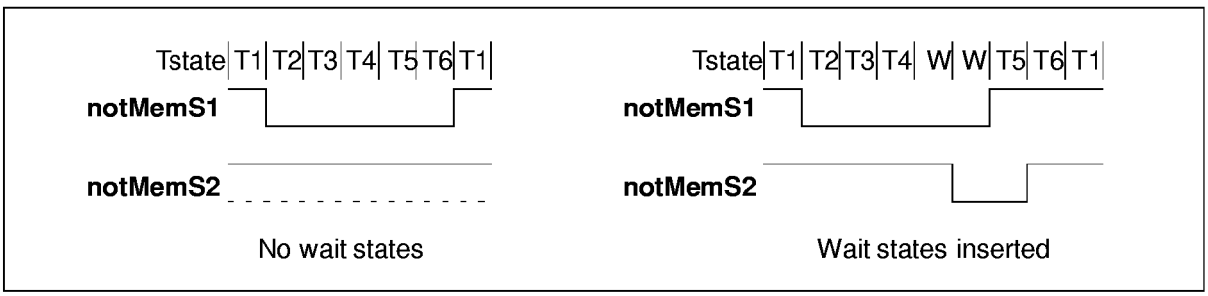


Figure 6.6 IMS T805 effect of wait states on strobes

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

Table 6.4 Strobe S0 to **ProcClockOut** skew

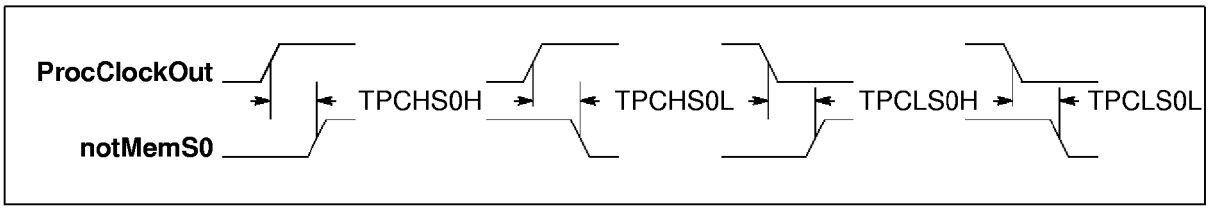


Figure 6.7 IMS T805 skew of **notMemS0** to **ProcClockOut**

6.3 Write cycle

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

For a write cycle pin **MemnotWrD0** will be low during **T1** and **T2**. Write data is placed on the bus at the start of **T3** and removed at the end of **T6**. If **T6** is extended to force the next cycle **Tmx** (page 18) to start on a rising edge of **ProcClockOut**, data will be valid during this time also.

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

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

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

Notes

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

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

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

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

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

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

Table 6.5 Write

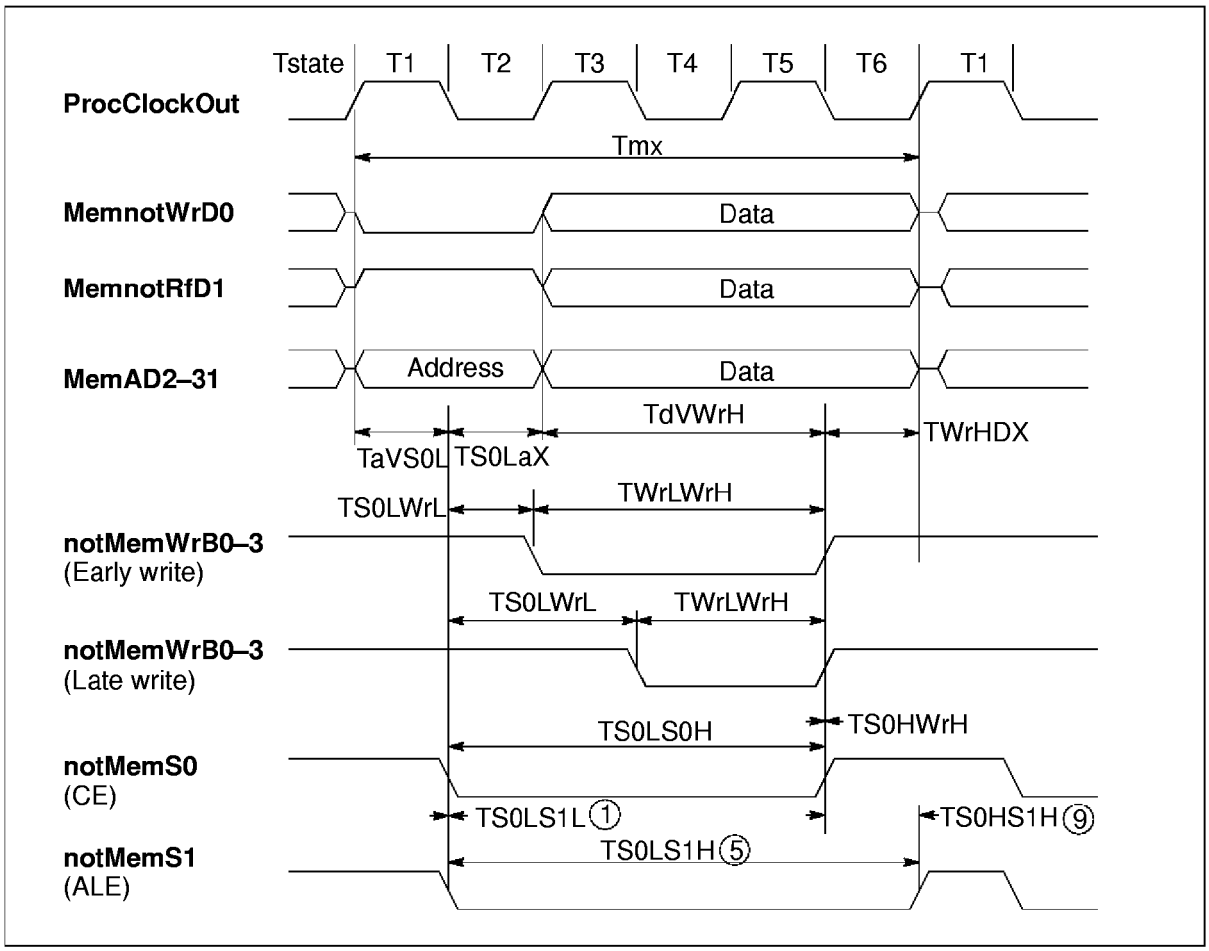


Figure 6.8 IMS T805 external write cycle

6.4 Wait

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

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

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

Symbol	Parameter	T805-25		Units	Notes
		Min	Max		
TPCLWtH	Wait setup	10		ns	1,2
TPCLWtL	Wait hold	8		ns	1,2
TWtLWtH	Delay before re-assertion of Wait	40		ns	

Notes

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

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

Table 6.6 IMS T805 memory wait

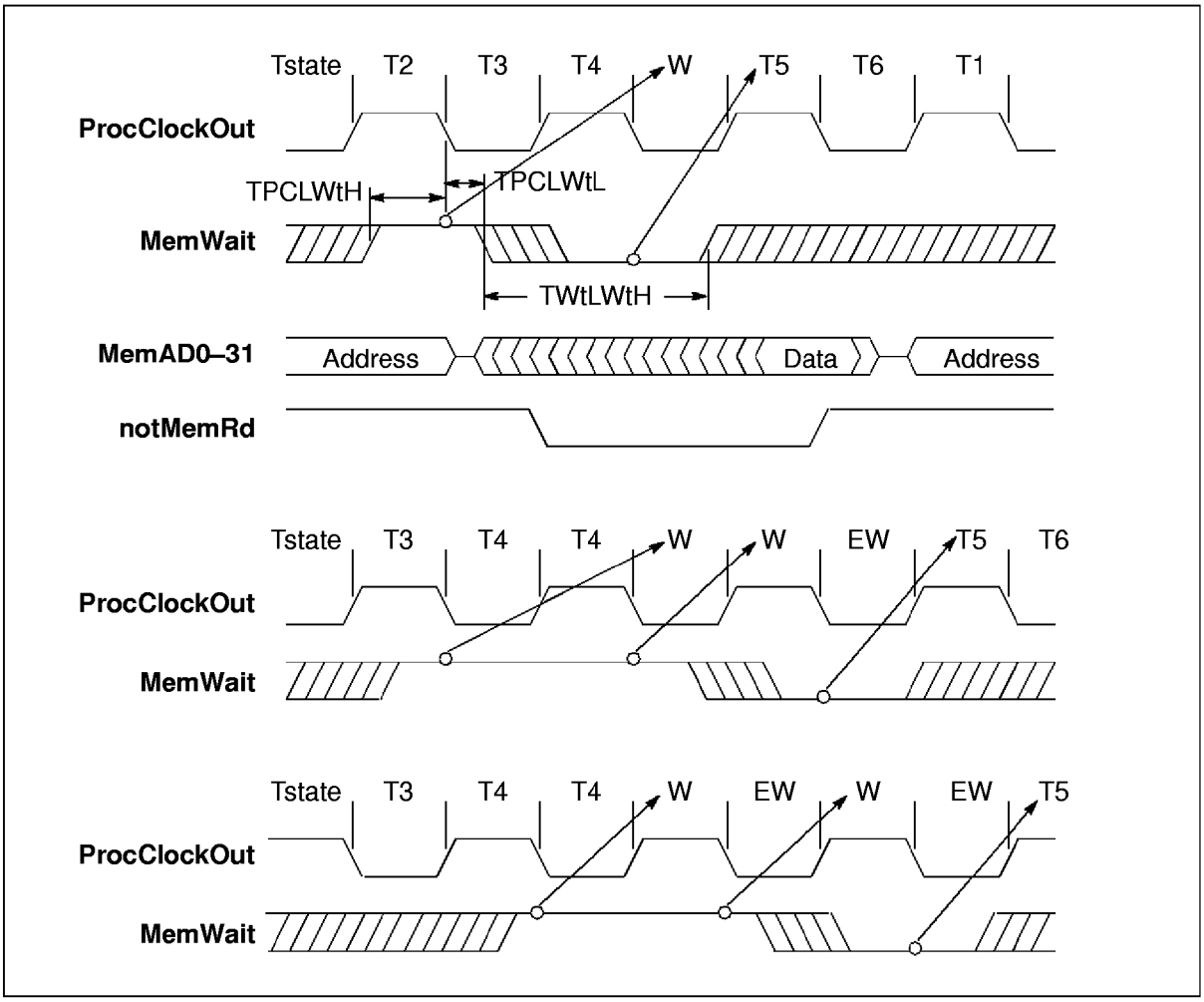


Figure 6.9 IMS T805 memory wait timing

6.5 Memory refresh

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

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

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

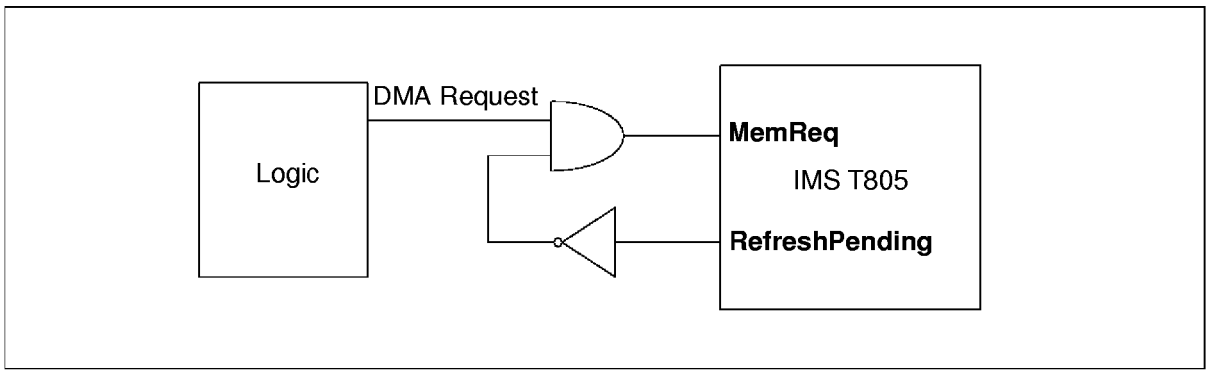


Figure 6.10 IMS T805 refresh with DMA

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

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

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

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

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

Notes

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

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

Table 6.7 Memory refresh

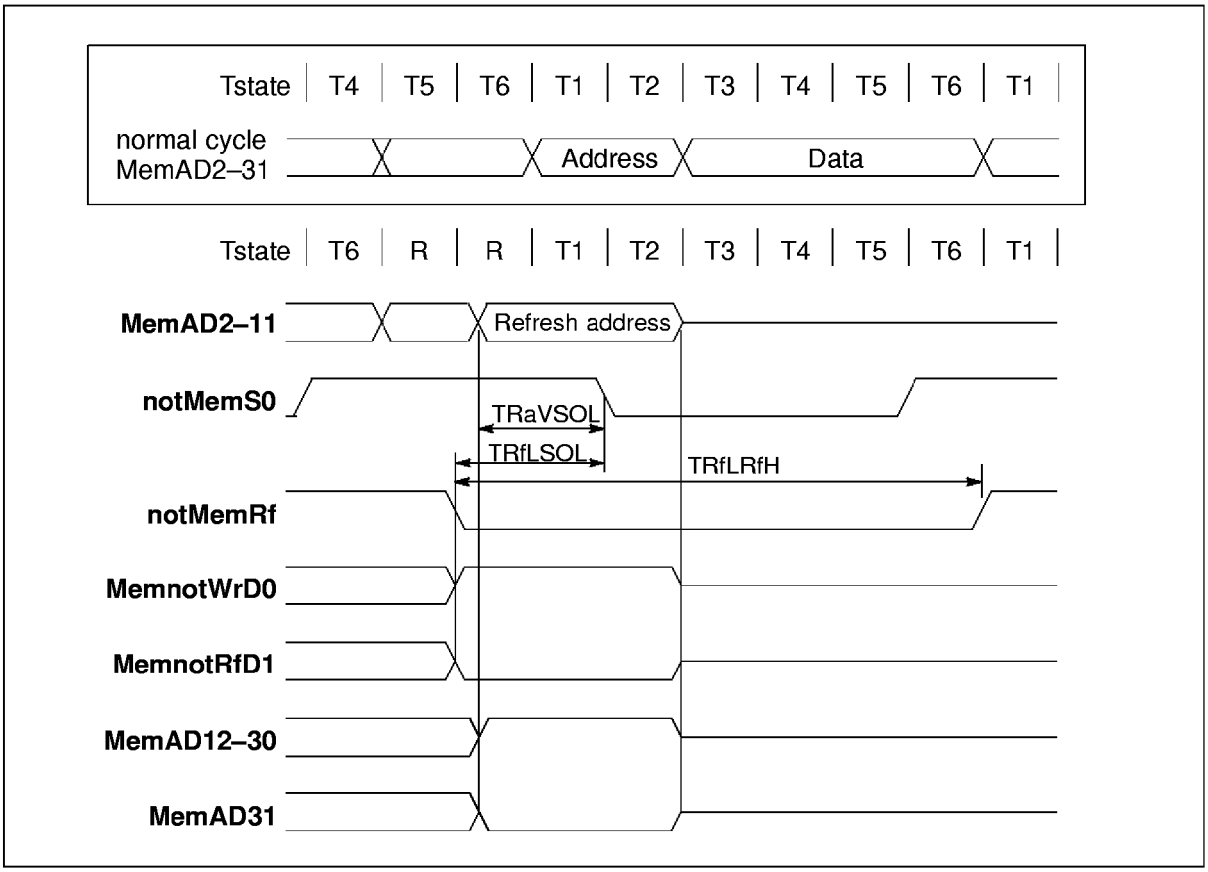


Figure 6.11 IMS T805 refresh cycle timing

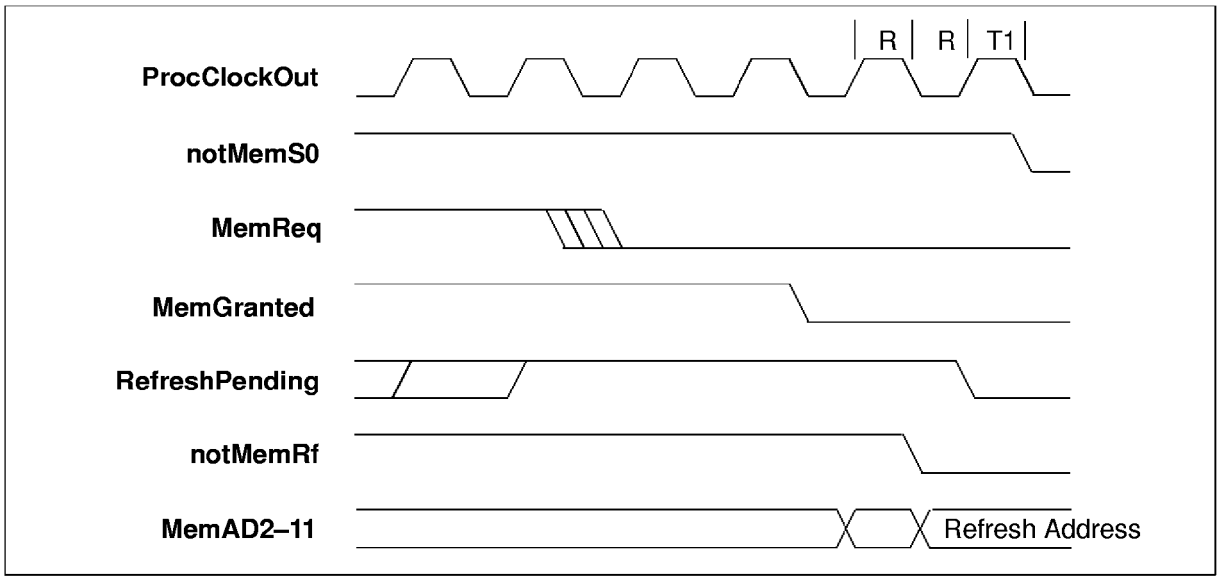


Figure 6.12 IMS T805 refresh pending timing diagram

6.6 Direct memory access

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

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

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

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

Symbol	Parameter	T805-25		Units	Note
		Min	Max		
TMRHPCL	**MemReq** setup before **ProcClockOut** falling	3	14	ns	1
TPCLMGH	**MemReq** response time	77	89	ns	2
TMRLPCL	**Memreq** removal before **ProcClockOut** falling	4	15		
TPCLMGL	**MemReq** end response time	40	54	ns	
TADZMGH	Bus tristate before **MemGranted**	0	22	ns	
TMGLADV	Bus active after end of **MemGranted**	0	26	ns	

Notes

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

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

Table 6.8 Memory request

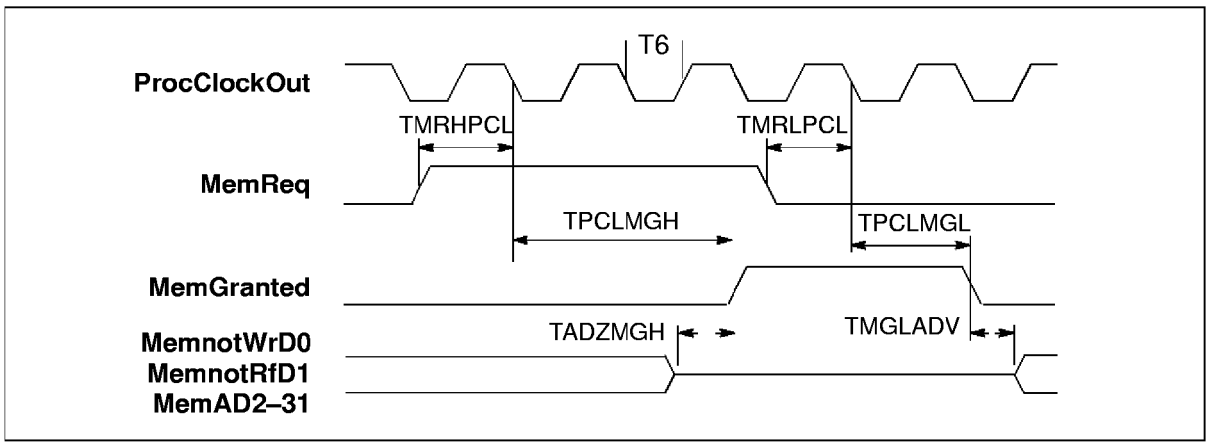


Figure 6.13 IMS T805 memory request timing

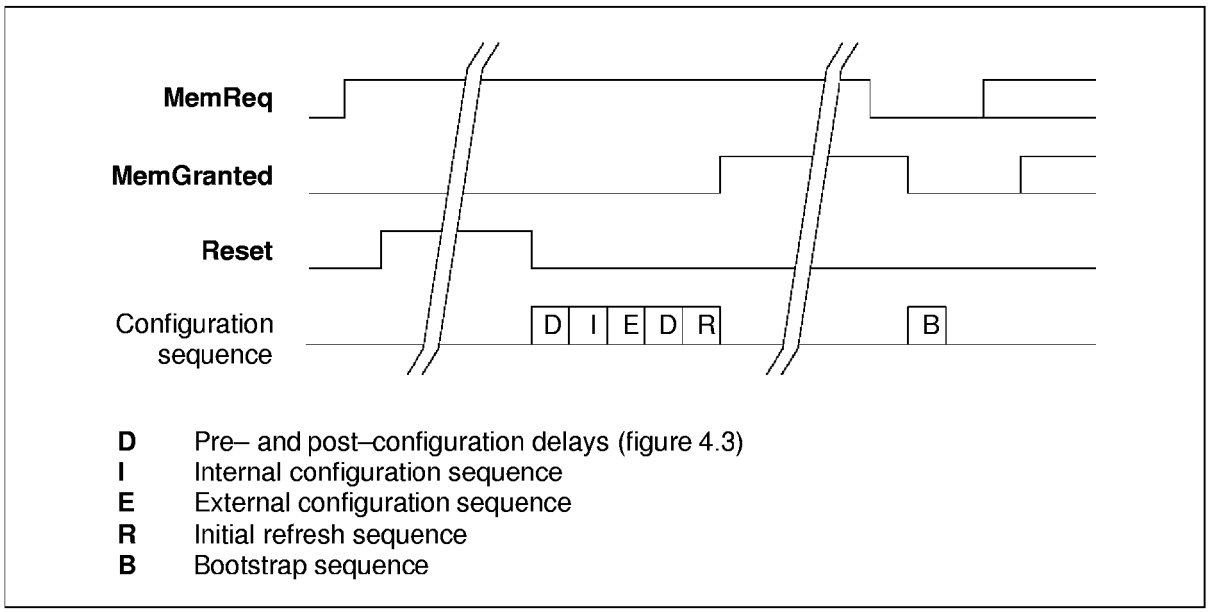


Figure 6.14 IMS T805 DMA sequence at reset

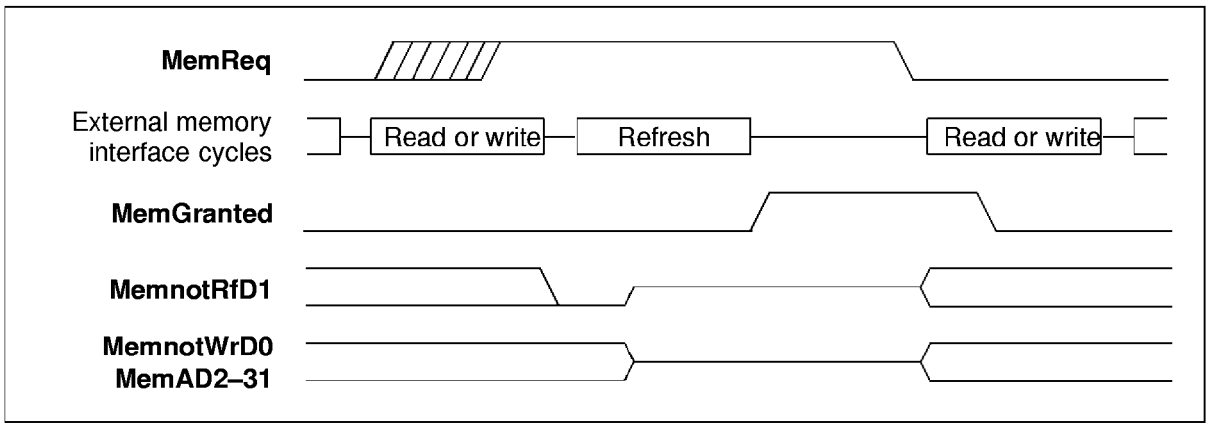


Figure 6.15 IMS T805 operation of **MemReq**, **MemGranted** with external, refresh memory cycles

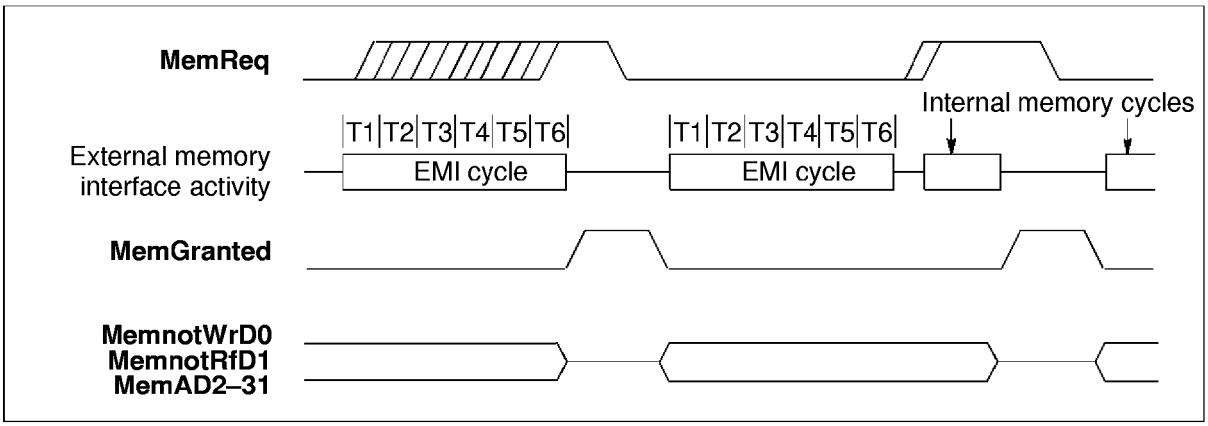


Figure 6.16 IMS T805 operation of **MemReq**, **MemGranted** with external, internal memory cycles

6.7 Memory configuration

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

6.7.1 Internal configuration

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

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

† Provided for static RAM only.

Table 6.9 IMS T805 internal configuration coding

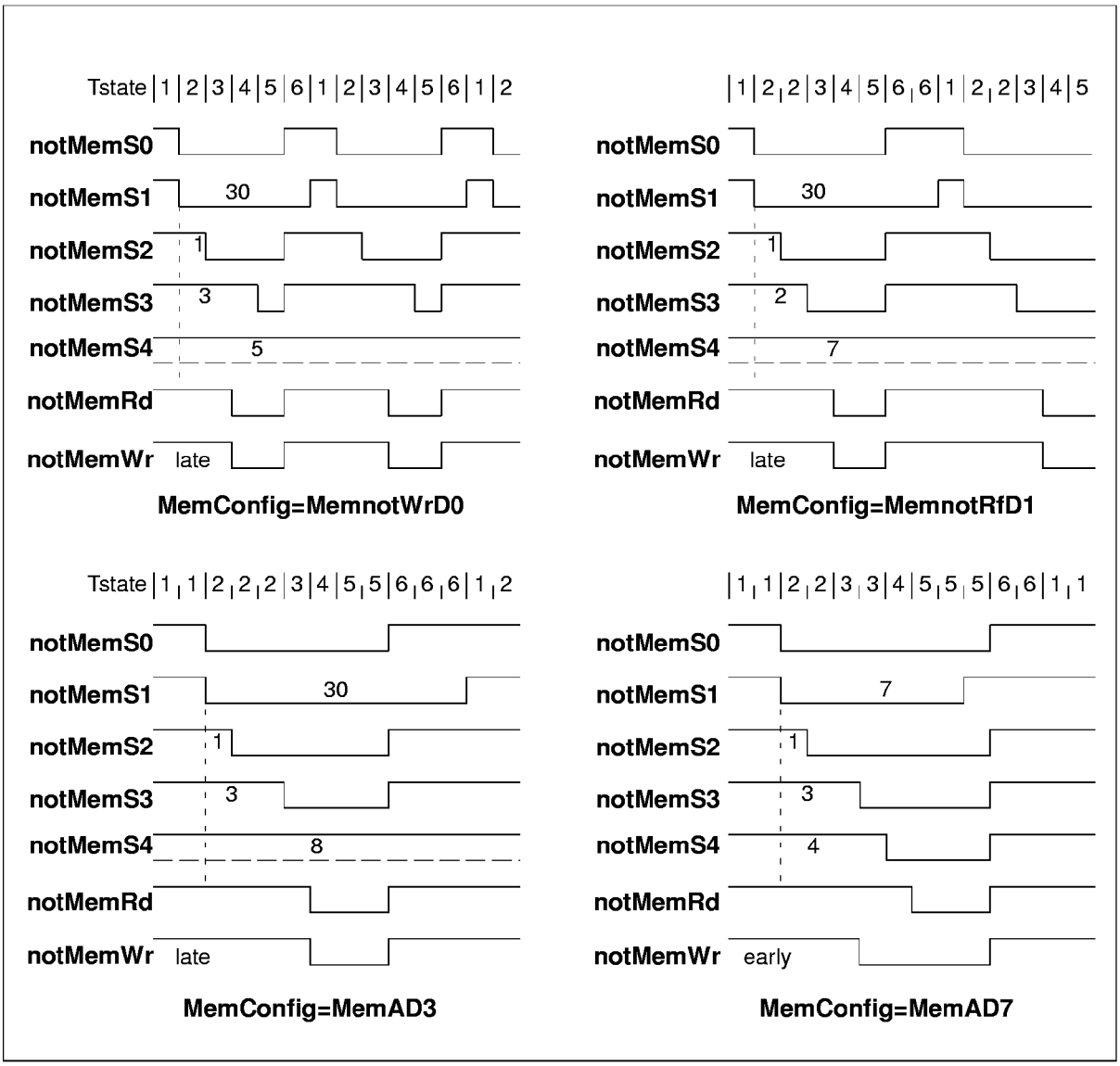


Figure 6.17 IMS T805 internal configuration

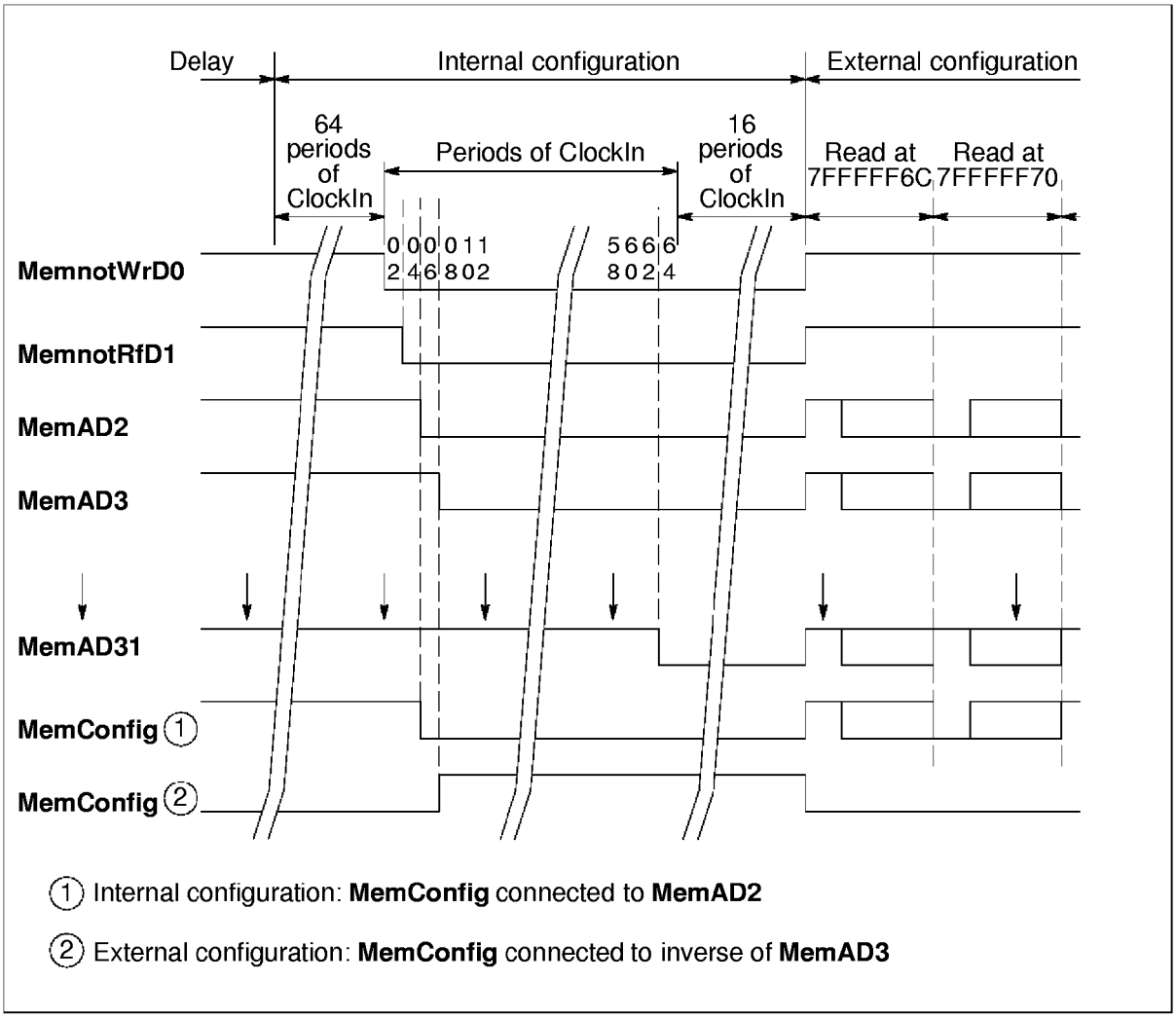


Figure 6.18 IMS T805 internal configuration scan

6.7.2 External configuration

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

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

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

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

The next five addresses (field 7) define the duration of **notMemS1** and the following fifteen (fields 8 to 10) define the delays before strobes **notMemS2-4** become active. The five bits allocated to each strobe allow durations of from 0 to 31 periods **Tm**, as described in strobes page 18.

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

The columns to the right of the coding table show the values of each configuration bit for the four sample external configuration diagrams. Note the inclusion of period **E** at the end of **T6** in some diagrams. This is inserted to bring the start of the next **Tstate T1** to coincide with a rising edge of **ProcClockOut** (page 19).

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

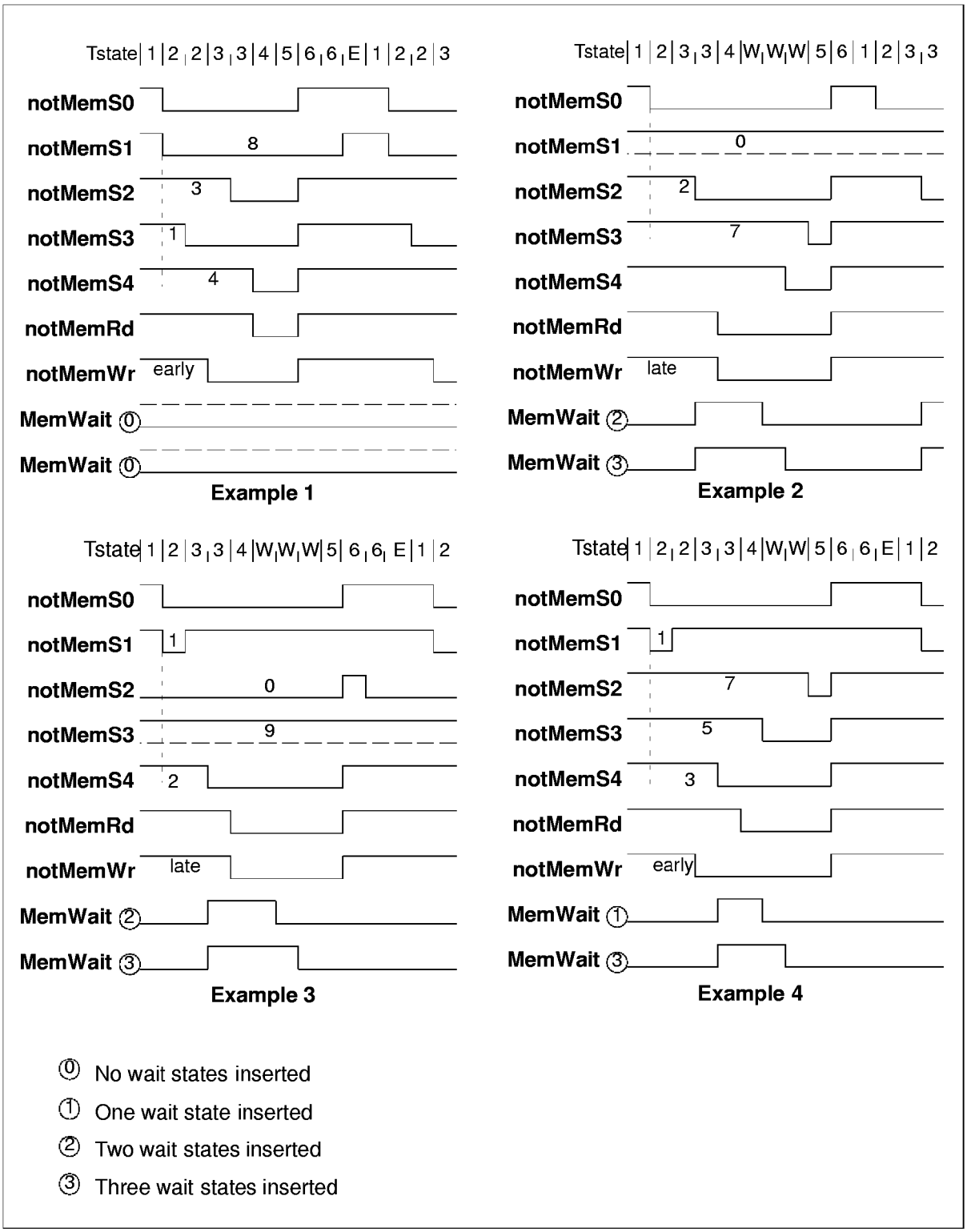
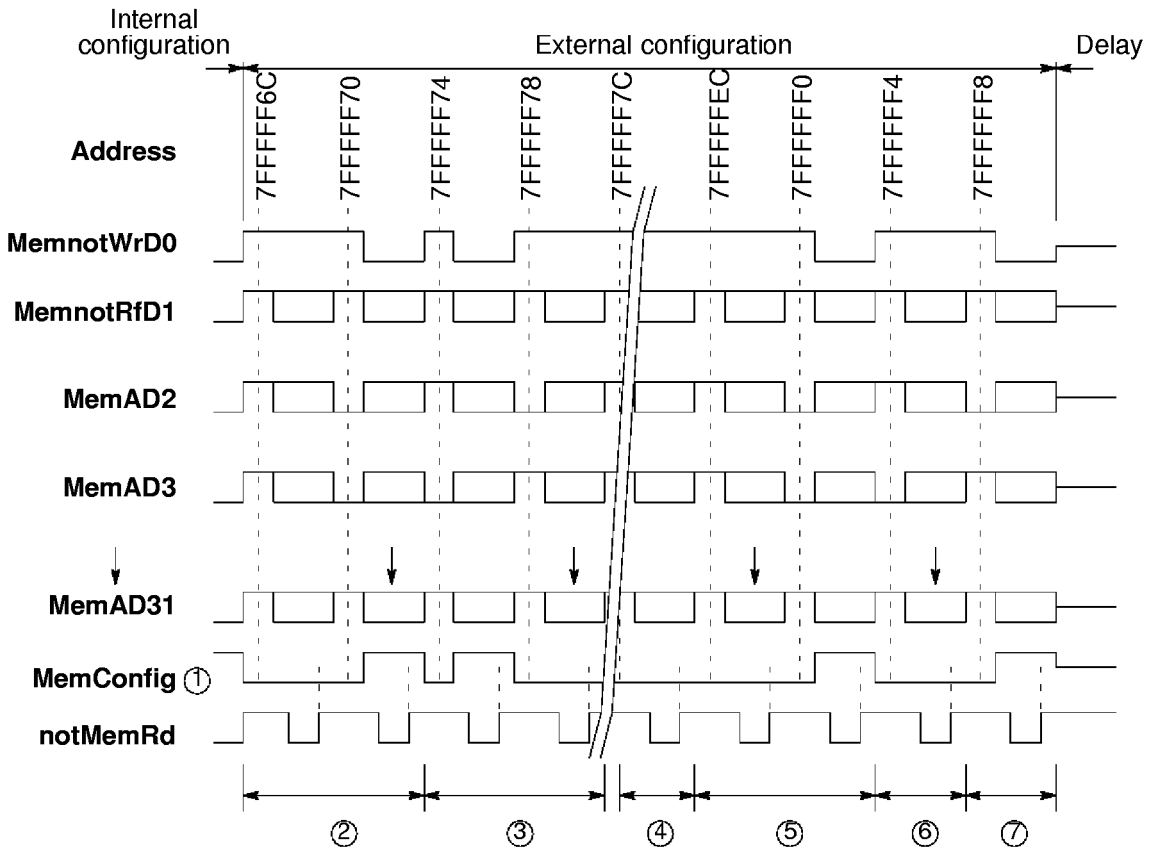


Figure 6.19 IMS T805 external configuration



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

Figure 6.20 IMS T805 external configuration scan

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

Table 6.10 IMS T805 external configuration coding

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

Table 6.11 IMS T805 memory refresh configuration coding

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

Interval = 18 * 200 = 3600 ns

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

Symbol	Parameter	T805-25		Units
		Min	Max	
TMCVRdH	**MemConfig** data setup	20		ns
TRdHMCX	**MemConfig** data hold	0		ns
TS0LRdH	**notMemS0** to configuration data read	308	332	ns

Table 6.12 Memory configuration

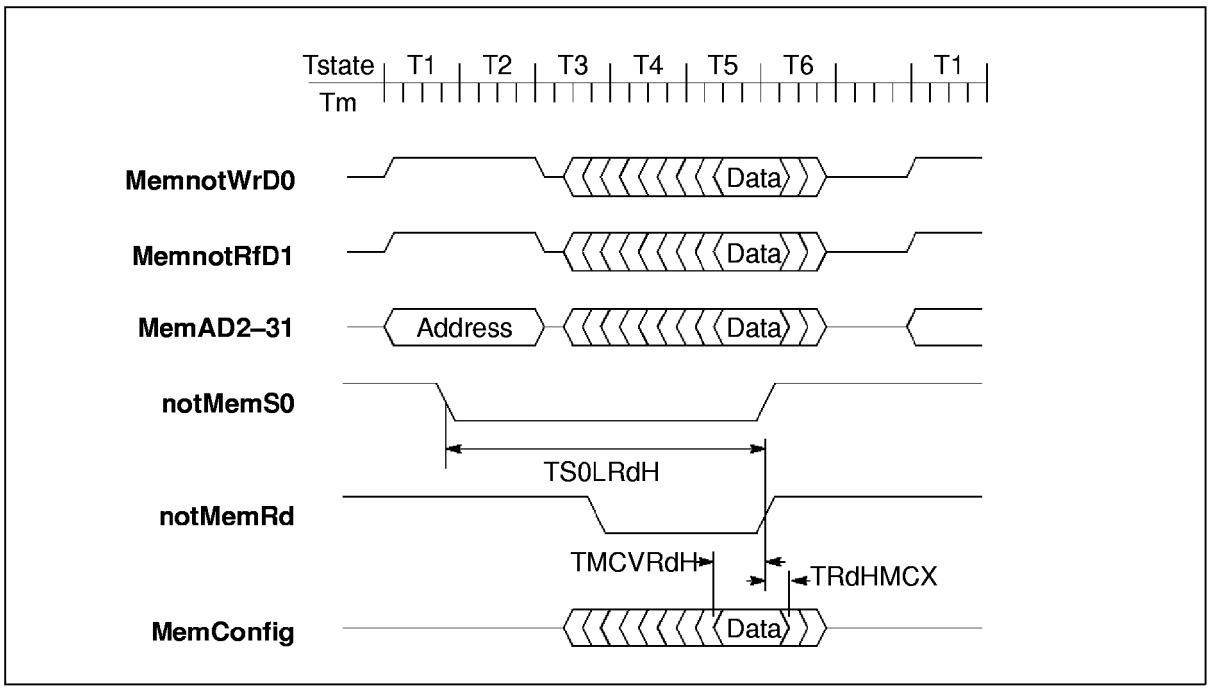


Figure 6.21 IMS T805 external configuration read cycle timing

7 Events

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

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

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

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

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

If the process is a high priority one and no other high priority process is running, typical latency is 19 full processor cycles **TCPLCPL** (38 **Tm**), and maximum latency (assuming all memory accesses are internal) is 78 full processor cycles (156 **Tm**) with the FPU in use, and 58 full processor cycles (116 **Tm**) with the FPU not in use. Setting a high priority task to wait for an event input allows the user to interrupt a transputer program running at low priority. The time taken from asserting **EventReq** to the execution of the microcode interrupt handler in the CPU is four cycles. The following functions take place during the four cycles:

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

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

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

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

Symbol	Parameter	T805-25		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
TKHEWL	**EventAck** to end of **EventWaiting**	0		ns

Table 7.1 Event

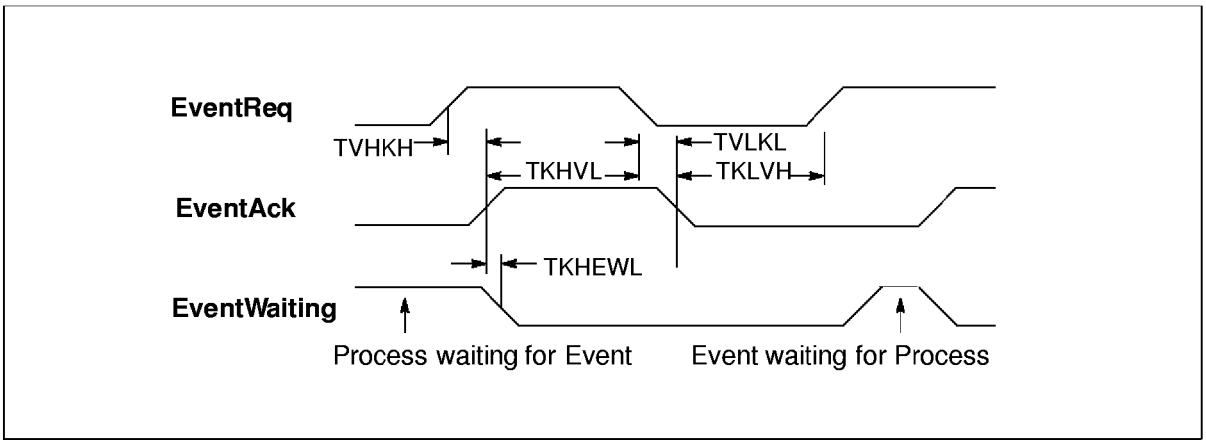


Figure 7.1 IMS T805 event timing

8 Links

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

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

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

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

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

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

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

Table 8.1 Speed Settings for Transputer Links

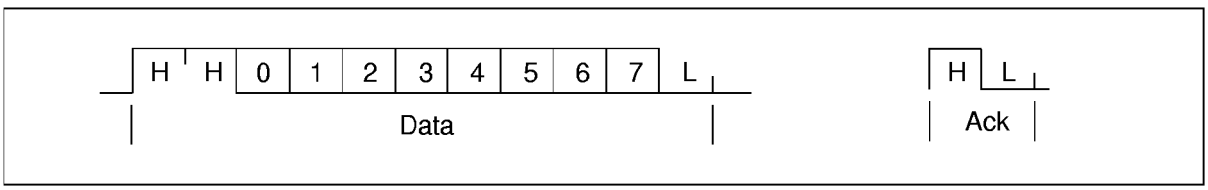


Figure 8.1 IMS T805 link data and acknowledge packets

Symbol	Parameter		Min	Nom	Max	Units	Notes
TJQr	**LinkOut** rise time				20	ns	
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 100 ohms 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 8.2 Link

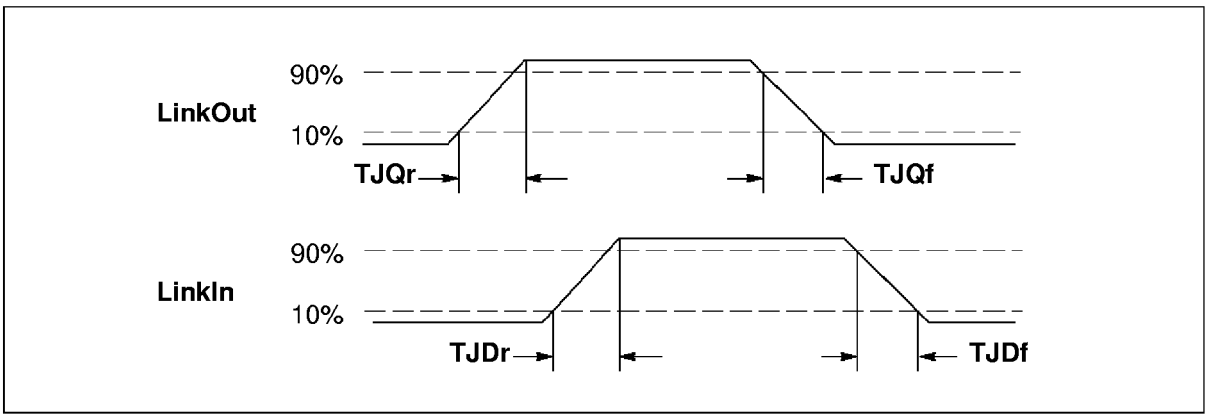


Figure 8.2 IMS T805 link timing

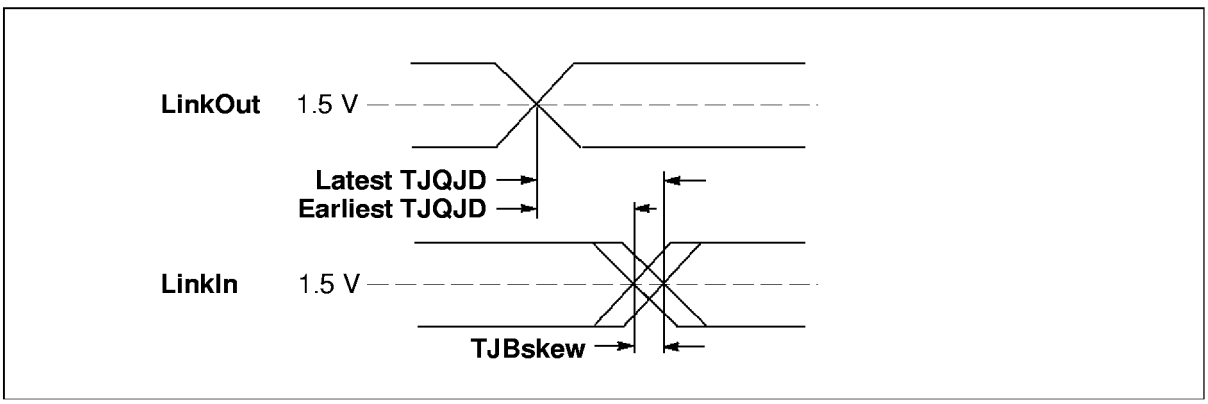


Figure 8.3 IMS T805 buffered link timing

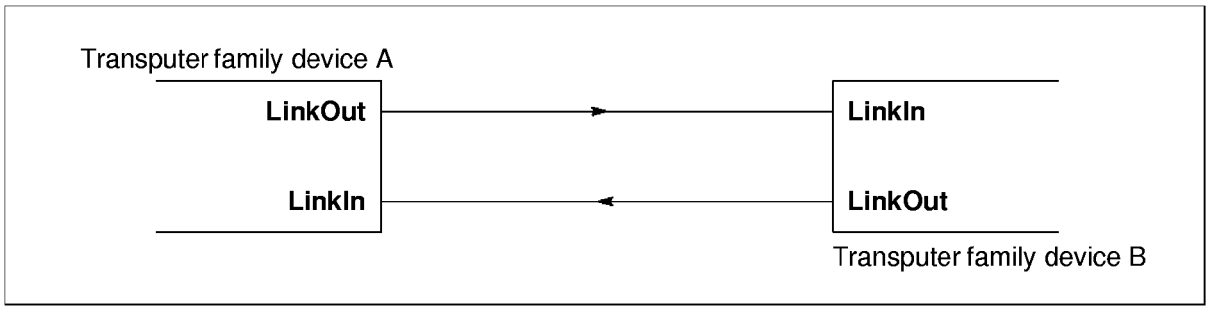


Figure 8.4 Links directly connected

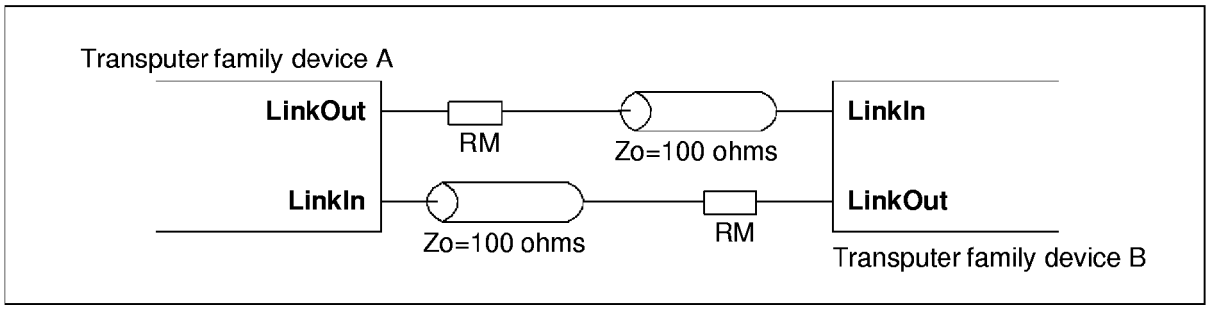


Figure 8.5 Links connected by transmission line

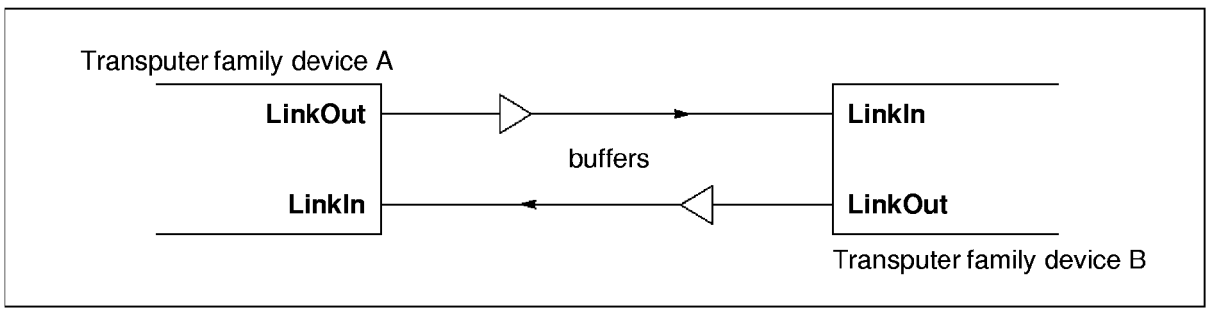


Figure 8.6 Links connected by buffers

9 Electrical specifications

9.1 DC electrical characteristics

SYMBOL	PARAMETER	MIN	MAX	UNITS	NOTES
VDD	DC supply voltage	0	7.0	V	1,2,3
V_I, V_O	Voltage on input and output pins	–0.5	VDD+0.5	V	1,2,3
I_I	Input current		±25	mA	4
t_{OSC}	Output short circuit time (one pin)		1	s	2
T_S	Storage temperature	–65	150	°C	2
T_A	Ambient temperature under bias	–55	125	°C	2
P_{Dmax}	Maximum allowable dissipation		2	W	

Notes

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

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

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

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

Table 9.1 Absolute maximum ratings

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

Notes

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

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

3 Excluding **LinkOut** load capacitance.

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

Table 9.2 Operating conditions

SYMBOL	PARAMETER		MIN	MAX	UNITS	NOTES
V_{IH}	High level input voltage		2.0	VDD+0.5	V	1, 2
V_{IL}	Low level input voltage		–0.5	0.8	V	1, 2
I_I	Input current	@ GND<VI<VDD		±10	µA	1, 2
V_{OH}	Output high voltage	@ IOH=2mA	VDD–1		V	1, 2
V_{OL}	Output low voltage	@ IOL=4mA		0.4	V	1, 2
I_{OZ}	Tristate output current	@ GND<V0<VDD		±10	µA	1, 2
P_D	Power dissipation			1.2	W	2, 3
C_{IN}	Input capacitance	@ f=1MHz		7	pF	
C_{OZ}	Output capacitance	@ f=1MHz		10	pF	

Notes

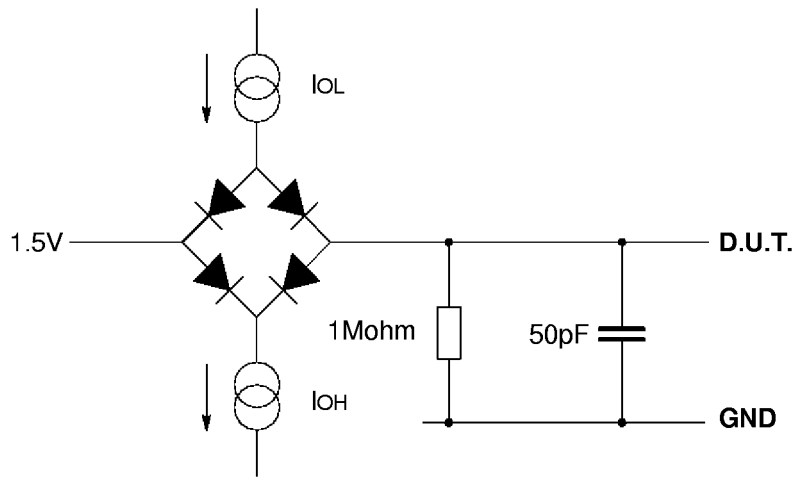
1 All voltages are with respect to **GND**.

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

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

Table 9.3 DC characteristics

9.2 Equivalent circuits



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

Figure 9.1 Load circuit for AC measurements

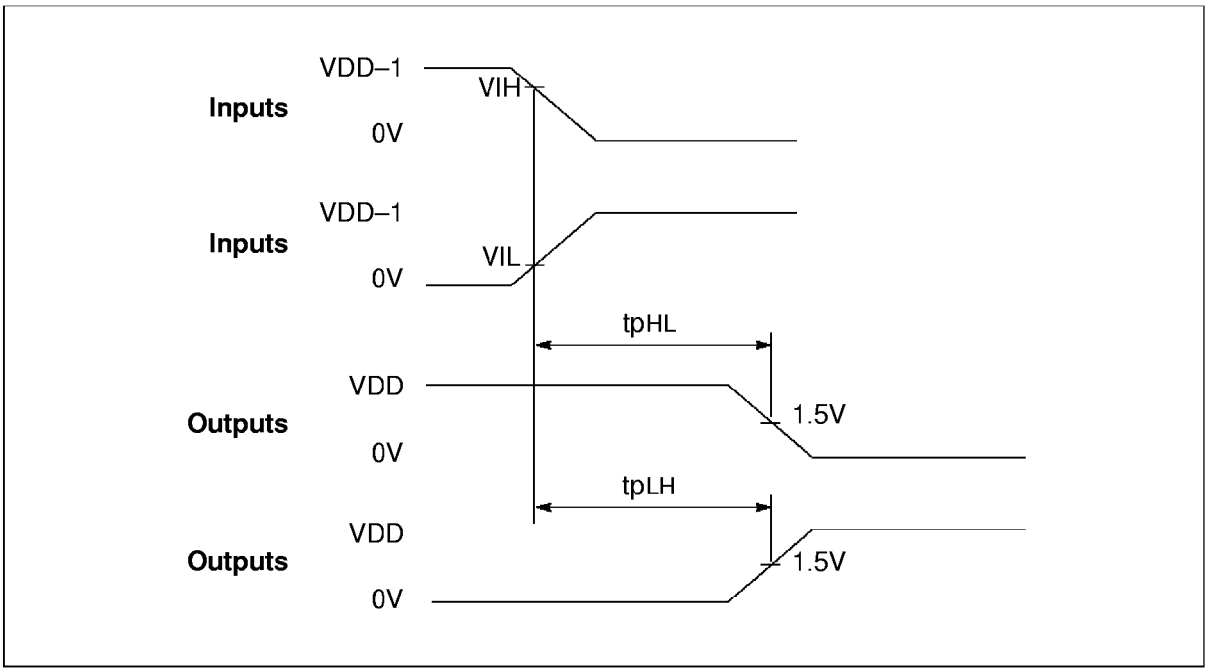


Figure 9.2 AC measurements timing waveforms

9.3 AC timing characteristics

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

Notes

1 Non-link pins; see section on links.

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

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

Table 9.4 Input and output edges

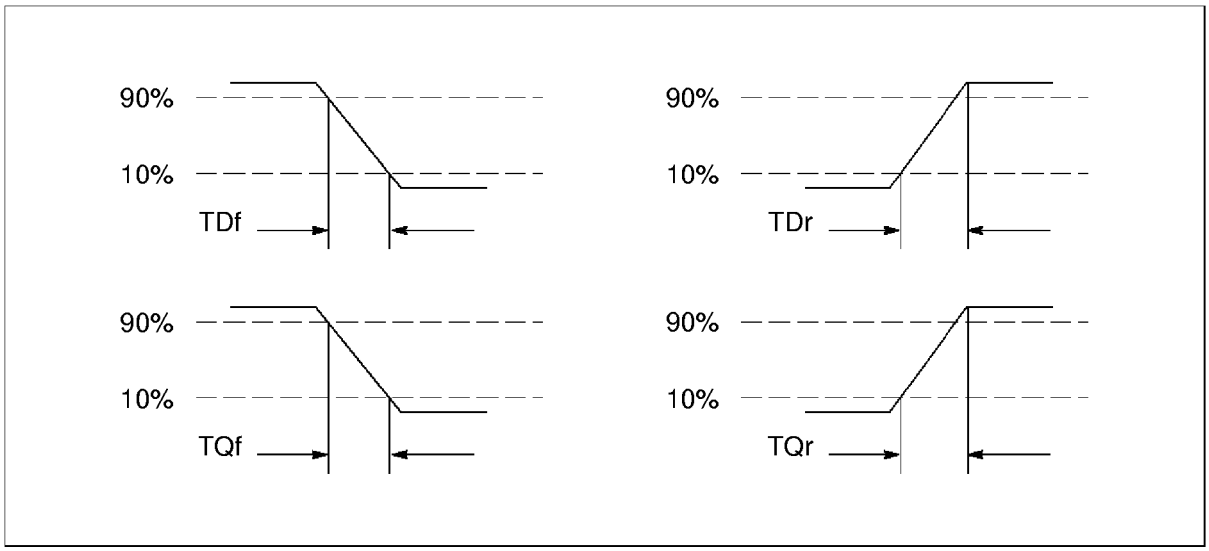


Figure 9.3 IMS T805 input and output edge timing

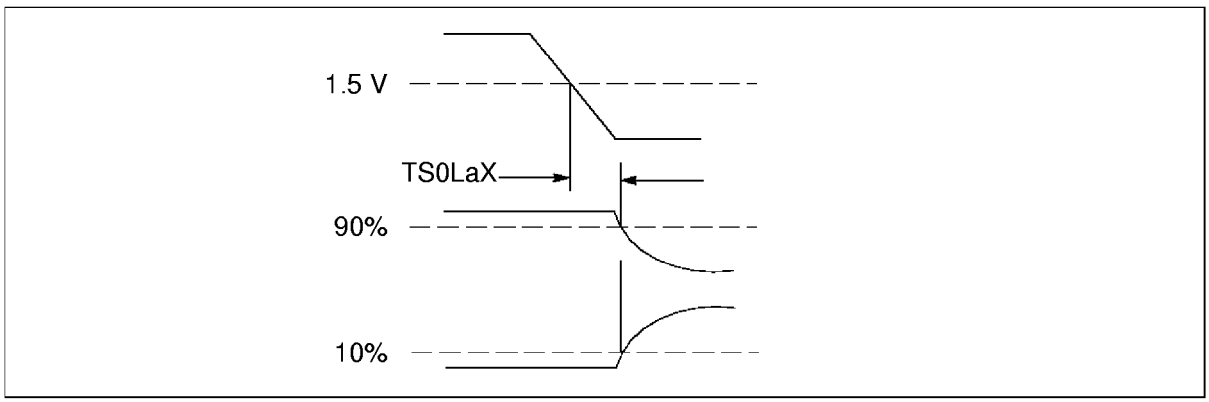
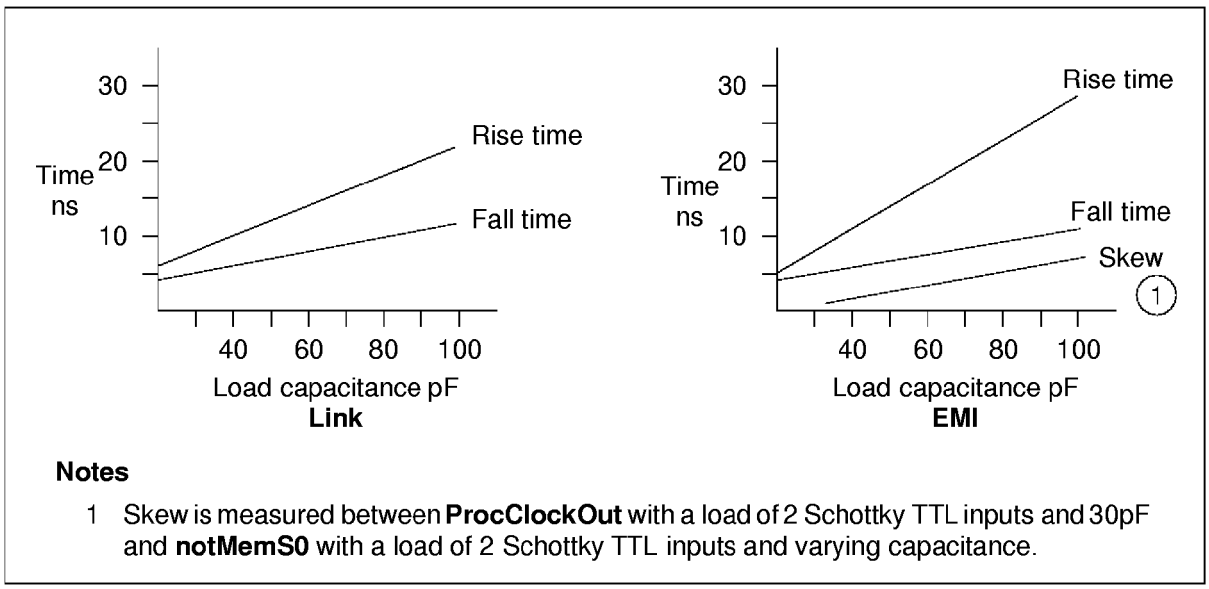


Figure 9.4 IMS T805 tristate timing relative to **notMemS0**



Notes

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

Figure 9.5 Typical rise/fall times

9.4 Power rating

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

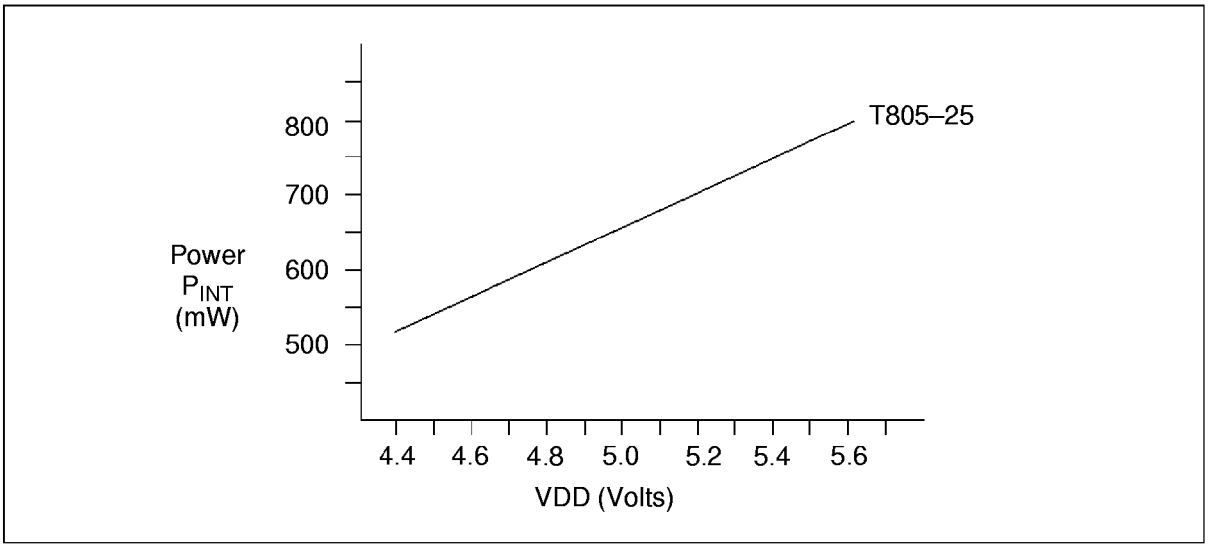


Figure 9.6 IMS T805 internal power dissipation vs VDD

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 10.3.

10 Package details

10.1 84 pin grid array package

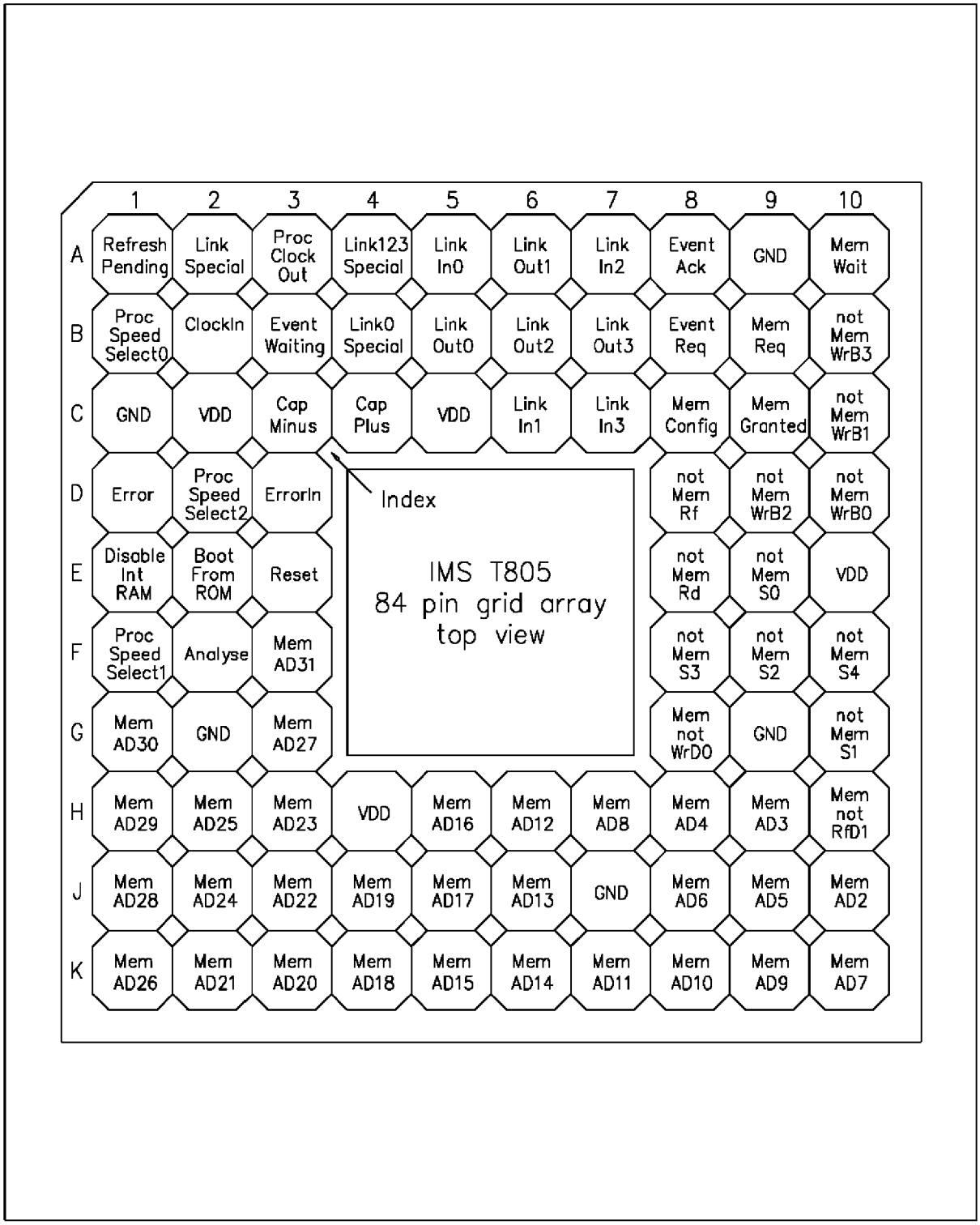


Figure 10.1 IMS T805 84-pin grid array package pinout

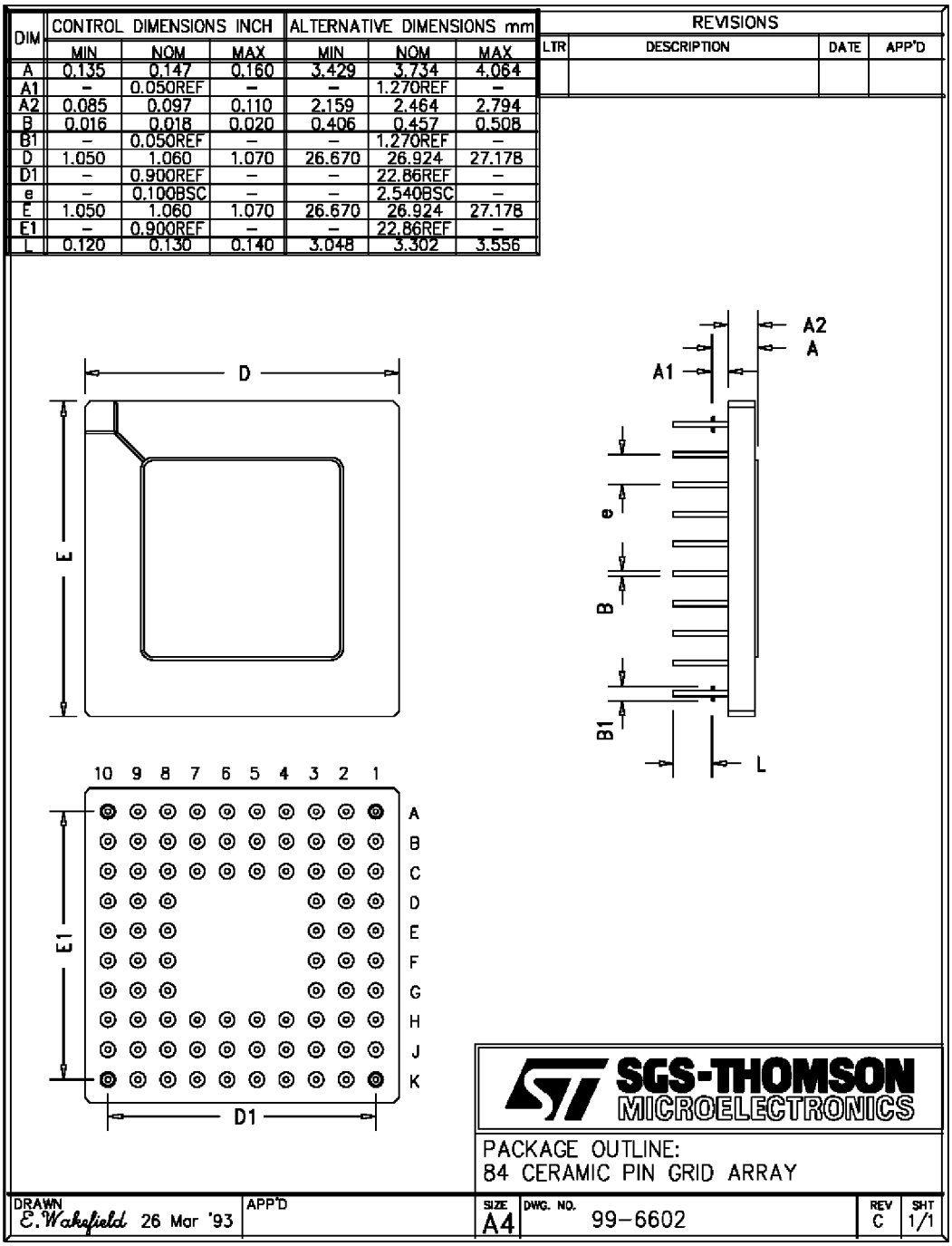


Figure 10.2 IMS T805 84-pin grid array package dimensions

10.2 100 pin cavity-down ceramic quad flat pack (CQFP) package

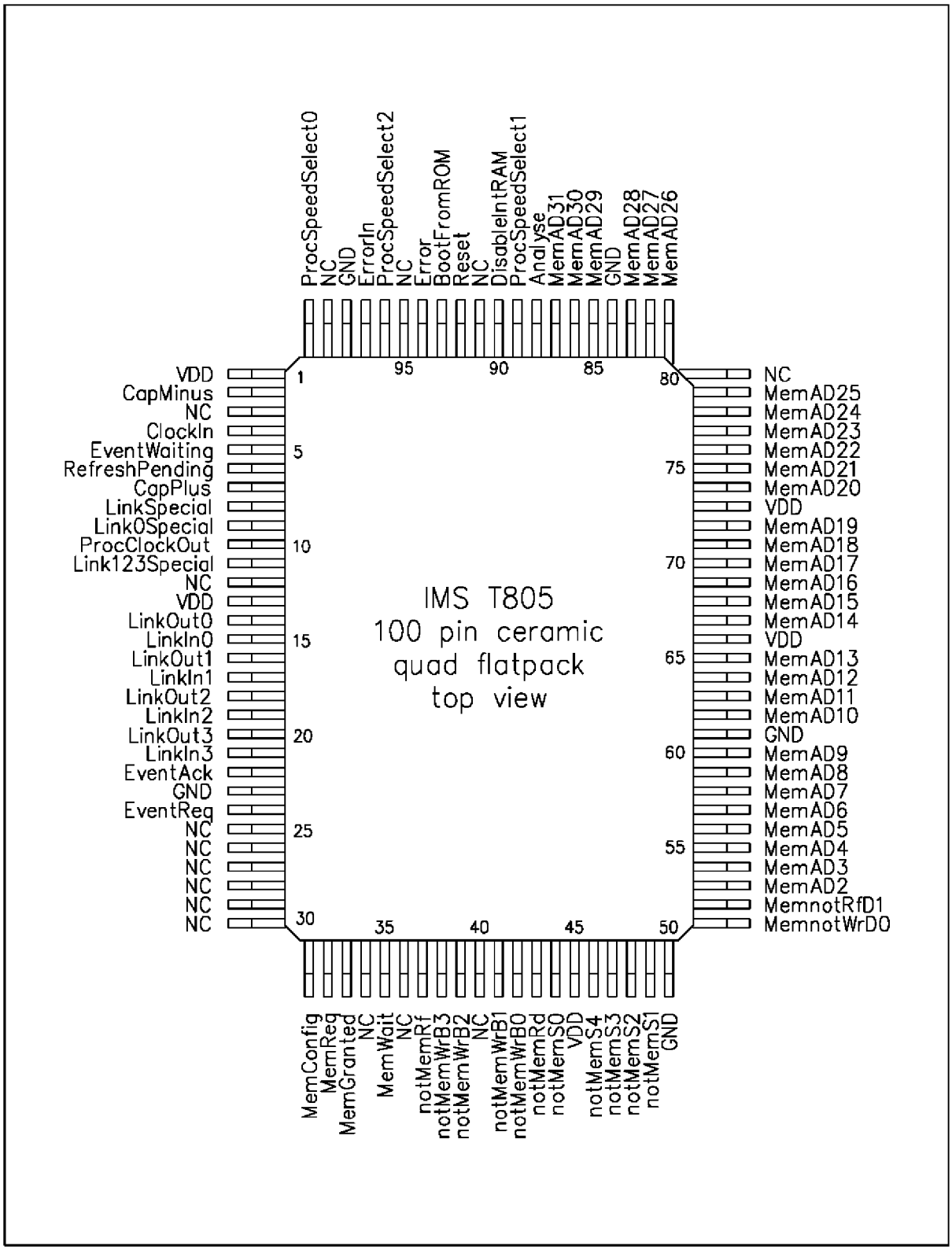


Figure 10.3 IMS T805 100-pin cavity-down ceramic quad flat pack (CQFP) package pinout

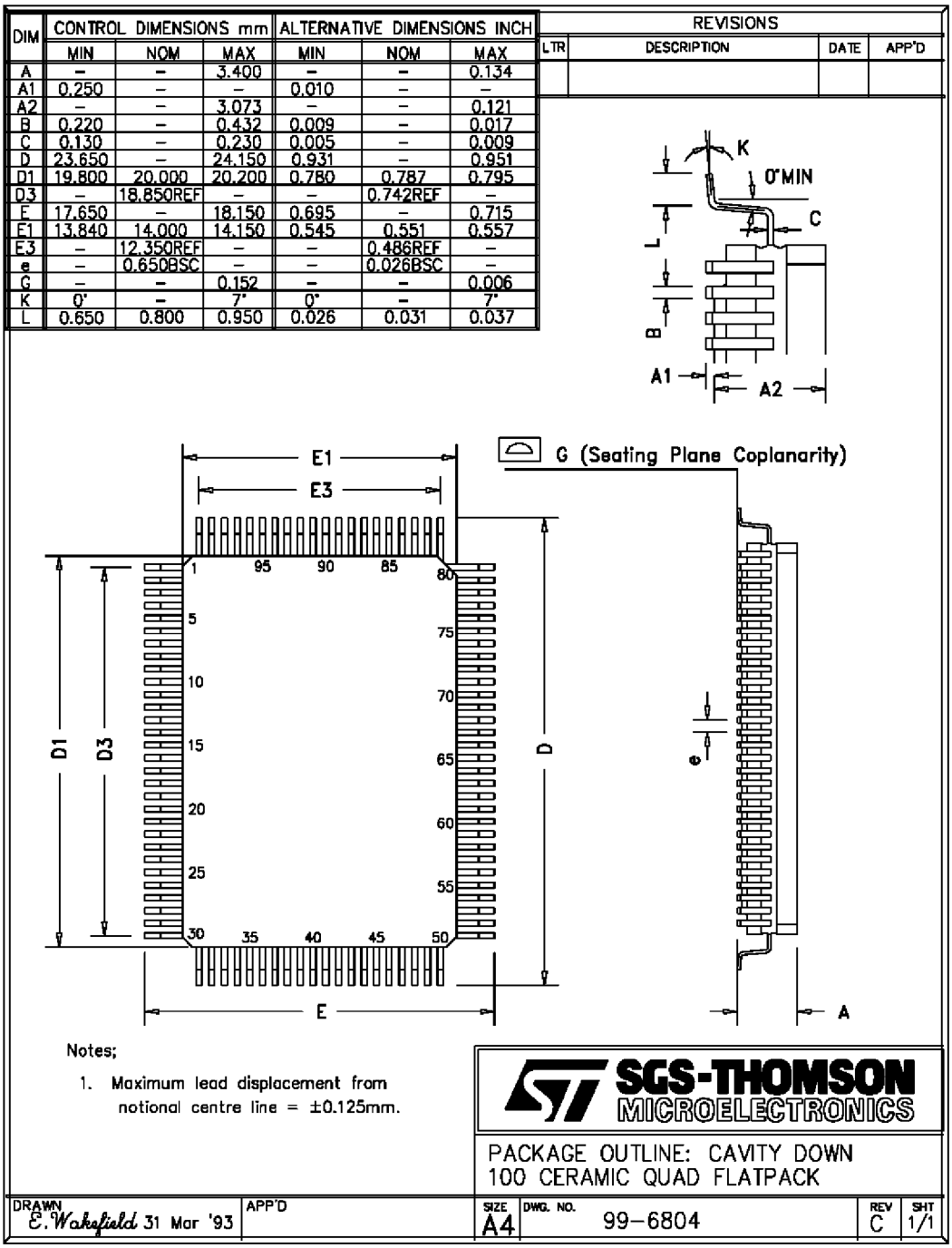


Figure 10.4 IMS T805 100-pin cavity-down ceramic quad flat pack (CQFP) package dimensions

10.3 Thermal specification

The IMS T805 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.

11 Ordering

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

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

SGS–THOMSON designation	Processor clock speed	Processor cycle time	PLLx	Package
IMS T805-G25S	25.0	40	5.0	84 pin ceramic pin grid array
IMS T805-F25S	25.0	40	5.0	100 pin ceramic quad flat pack

Table 11.1 IMS T805 ordering details

An extended temperature version is available, see the *IMS T805E Datasheet* for details.

12 Transputer instruction set summary

12.1 Introduction

The Function Codes table 12.9 (page 63) 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 12.1.

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 12.1 *prefix* coding

Tables 12.10 to 12.30 (pages 63–70) 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 12.2 *operate* coding

12.1.1 Product identity numbers

The load device identity (*lddevid*) instruction (table 12.10) 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 12.3.

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 12.3 Product identity numbers

12.1.2 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 12.23 to 12.29, 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*.

12.1.3 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 12.4.

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 12.4 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	12.2
E	The instruction will affect the **Error** flag	12.3
F †	The instruction will affect the **FP_Error** flag	12.6
† applies to IMS T805 only		

Table 12.5 Instruction features

12.2 Descheduling points

The instructions in table 12.6 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 12.6 Descheduling point instructions

12.3 Error instructions

The instructions in table 12.7 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 12.6), 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 12.7 Error setting instructions

12.4 Debugging support

Table 12.20 (page 67) 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.

12.5 Block move

The block move instructions (Table 12.21) 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.

12.6 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 12.8 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 12.8 Floating point error setting instructions

12.7 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 12.9 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 12.10 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 12.11 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 12.12 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 12.13 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	2**w**+8	2**w**+8	move message	
† does not apply to IMS T225						

Table 12.14 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 12.15 Timer handling operation codes

Operation Code	Memory Code	Mnemonic	Processor Cycles	Name	DEF
07	F7	in	2**w**+19	input message	D
0B	FB	out	2**w**+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 12.16 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 12.17 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 12.18 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 12.19 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 12.20 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 12.21 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 12.22 CRC and bit operation codes

12.8 Floating point instructions

12.9 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 12.23 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 12.24 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 12.25 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 12.26 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 12.27 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 12.28 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 12.29 Floating point arithmetic operation codes

12.10 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 12.30 Floating point support operation codes