IMS T424 transputer

<u>IMS T424:</u> 32 bit system providing 10 MIPS (million instructions per second) processing power with memory and concurrent communication capability all on a single chip.

<u>Processor:</u> Instruction set designed for high performance compact programs, efficient high level language implementation and direct support of concurrency.

<u>Memory:</u> 4K bytes of fast static RAM. Multiport access for processor, memory interface and INMOS serial links.

<u>Memory Interface:</u> 32-bit multiplexed interface with configurable timing to support all types of memory. Direct access of up to 4G bytes of memory with a maximum data rate of 25M bytes/s.

<u>INMOS Serial Links:</u> Four serial links provide concurrent message passing to other transputers. Supports a standard rate of 10M bits/s full duplex on each link.

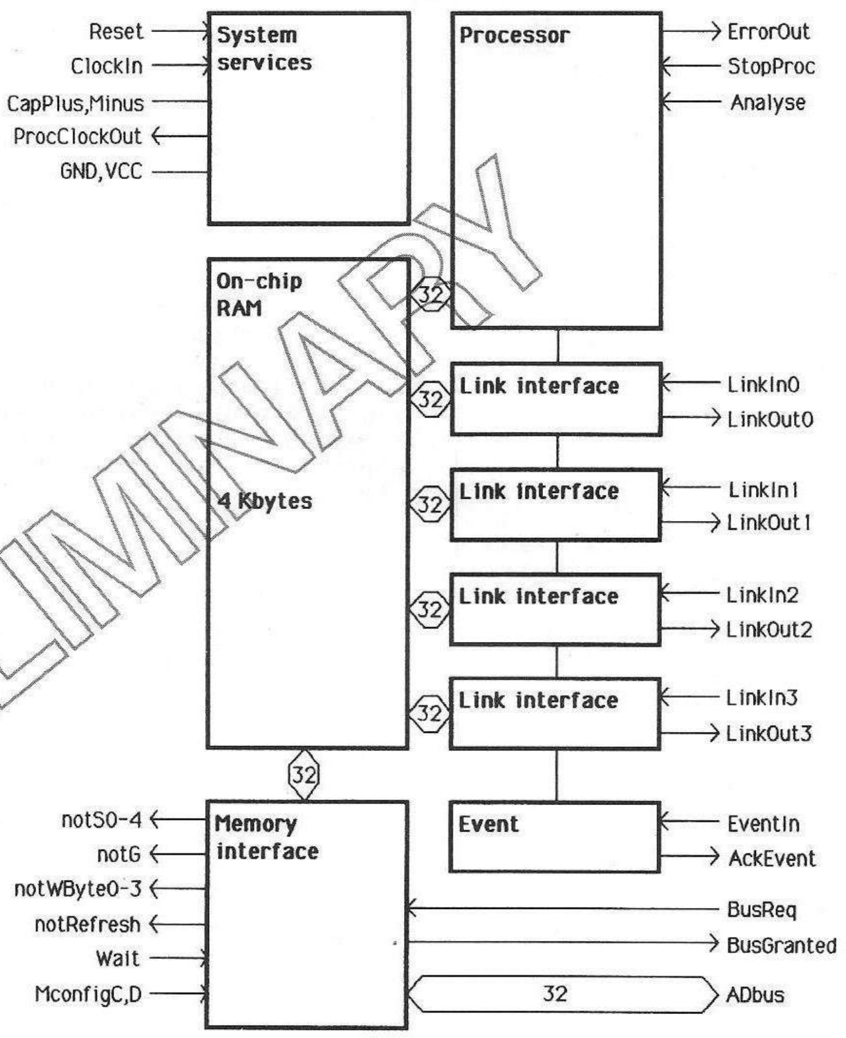
<u>Programming:</u> Programmable in a variety of high level languages. Occam support currently available.

<u>Engineering:</u> 5MHz external clock for simple engineering of transputer systems.

<u>Technology:</u> 250,000 devices fabricated in an advanced 2 micron CMOS process. Dissipates less than 1 watt. Packaged in an 84 pin ceramic leadless chip carrier.

<u>Family Range:</u> INMOS will offer a complete family of transputer products. Initial members will include a 16-bit transputer, a graphics processor and a disk processor.

IMS T424 block diagram



The Transputer Architecture

The transputer architecture defines a family of programmable VLSI components.

The transputer, itself, can be used as a stand-alone component or in networks to build high performance concurrent systems. As a stand-alone component, the transputer can be programmed in any of the conventional high level languages.

A typical transputer product contains processor, memory and communication links which provide point to point connection to other transputers. In addition, each transputer product also contains special circuitry and interfaces adapting it to a particular use, such as a memory interface or graphics control.

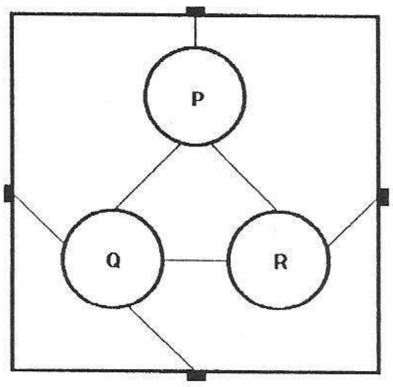
Occam provides a methodology for designing concurrent systems using transputers in just the same way that boolean algebra provides a methodology for designing today's electronic systems from logic gates.

The assembly level langage for the transputer is the programming language occam. It provides the efficiency in terms of program density and performance of an assembler, while offering the productivity and reliability advantages of programming in a high level language.

The system designers' task is eased because of the architectural relationship between occam and the transputer. The transputer has special features to implement occam, while a program executing in a transputer is formally equivalent to an occam process, so that a network of transputers can be described directly as an occam program.

Occam model of concurrency

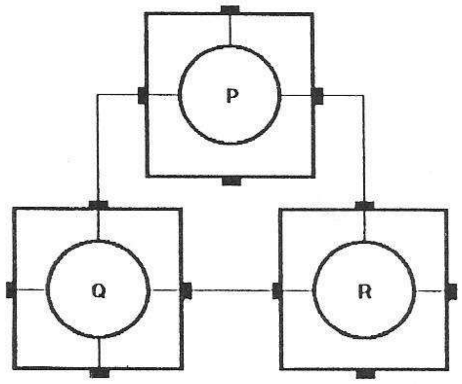
In occam, processes are interconnected to form concurrent systems. Each process can be thought as a black box with internal state, which can communicate with other processes using point to point communication channels. Processes can be used to represent many things, for example, a logic gate, a microprocessor,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



A program on a single transputer

The processes themselves are finite. Each process starts, performs a number of actions and then terminates. An action may be a set of sequential processes performed one after another, as in a conventional programming language, a set of parallel processes to be performed at the same time as one another, or a set of alternative processes, only one of which is performed, depending on some criterion.

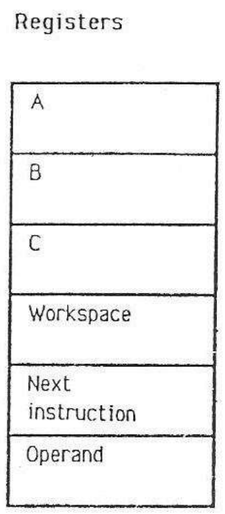
Communication between processes occurs when both processes are ready to communicate. This models the handshake method of communication used in hardware systems. Since the process may have internal concurrency, it may have many input channels and output channels performing communication at the same time.



The same program on three transputers

Occam can be used to program an individual transputer or to program a network of transputers. When occam is used to program an individual transputer, the transputer shares its time between the concurrent processes and channel communication is implemented by block moves within the memory. When occam is used to program a network of transputers, each transputer executes processes and the interprocess communication is implemented directly by transputer links. The same occam program can be implemented on a variety of transputer configurations, enabling tradeoffs to be made between costs, performance and response time.

The IMS T424 Processor



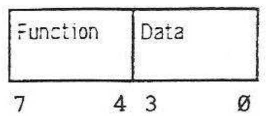
The processor of the T424 is designed to implement high level languages efficiently, with particular support for concurrent processes. Concurrency is supported by hardware scheduling mechanisms, enabling any number of processes to be executed together, sharing the processor. The processes communicate via efficient message transfer instructions using memory to memory block moves which fully utilize the bandwidth available from the on-chip RAM.

Context switches are fast because of the small number of registers in the process context, the dedicated scheduling instructions, and the high speed internal RAM.

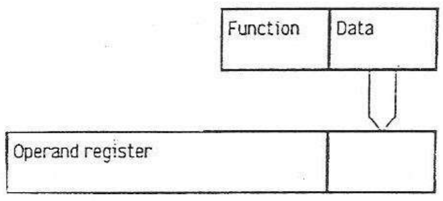
The T424 processor supports two levels of process priority. High priority processes may be used both for message through-routing or for fast response to external events for interrupt servicing.

The processor includes a timer which allows each process in the transputer to have its own view of time.

Occam



Instruction format



Although the T424 may be used with many high-level languages, optimal performance is gained by programming in occam. Occam is an effective language on any computer but particularly on the transputer.

Occam allows systems to be described as a collection of concurrent processes which communicate using channels. An occam program may be executed on a network of interconnected computers, each executing one of the concurrent processes. However with no changes except to configuration details it may be implemented on any smaller network or on a single computer, with each computer shared between a set of concurrent processes.

Programs are constructed from processes combined together using constructors. The primitive processes of input, output and assignment form the lowest level of process in a program.

The T424 instruction set supports the occam primitives directly. The list of occam keywords is given below.

Input and Output
```
   InputChannel ? char
   OutputChannel ! char
```

Sequential operations
```
   SEQ
   IF
   WHILE x > Ø
```

Parallel Operations
```
   PAR
   ALT
```

Timer
```
   TIME ? (>) e
```

Abstraction
```
   PROC echo (CHAN in,out)=
```

Execution of Sequential Programs

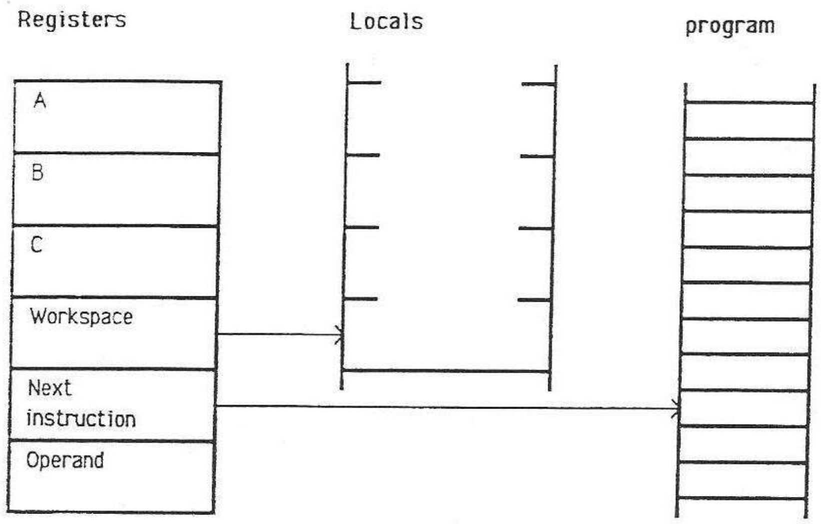
Sequential programs use the following registers:

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

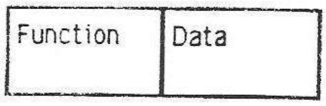
The instruction pointer which points to the next instruction to be executed.

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

The evaluation stack of three registers, A, B and C The evaluation stack is used for expression evaluation, to hold the operands of scheduling and communication instructions, and to hold the first three parameters of procedure calls.



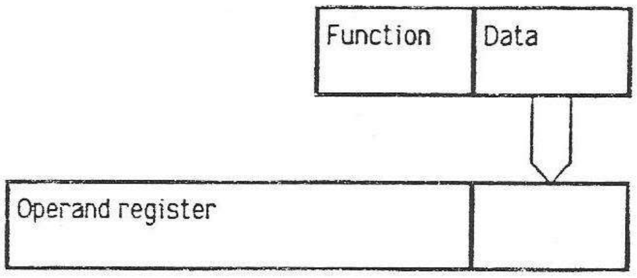
Instructions are fetched and executed using the instruction pointer. All instructions have the same format. Each is one byte long, and is divided into two 4-bit parts. The four most significant bits of the byte are the function code, and the four least significant bits are the operand.



This representation provides for sixteen functions, each with an operand ranging from Ø to 15. Thirteen of these values are used to encode the most frequently occurring functions.

Two more of the function codes, 'prefix' and 'negative prefix', are used to allow the operand of any instruction to be extended in length. All instructions start by loading the four data bits into the least significant four bits of the operand register, which is then used as the instruction's operand. All instructions, except 'prefix' and 'negative prefix', end by clearing the operand register, ready for the next instruction.

The 'prefix' instruction loads its four data bits into the operand register, and then shifts the operand register up four places. The 'negative prefix' instruction is similar, except that it complements the operand register before shifting it up. Consequently operands can be extended to any length up to the length of the operand register by a sequence of prefixing instructions.



The processor does not allow interrupts between a prefixing instruction and the following instruction. This removes the need to save the operand register on an interrupt.

The remaining function code, 'operate', causes its operand to be interpreted as an operation on the values held in the evaluation stack.

The 'operate' instruction allows up to 16 such operations to be encoded in a single byte instruction. However, the prefixing instructions can be used to extend the operand of an operate instruction just like any other. This allows the number of operations in the machine to be extended almost indefinitely.

Loading a value onto the evaluation stack pushes B into C, and A into B, before loading A. Storing a value from A, pops B into A and C into B.

The A, B and C registers are the sources and destinations for arithmetic and logical operations. For example, the 'add' instruction adds the A and B registers, places the result in the A register, and copies C into B.

In the following examples, the variables x, y and z are assumed to be in the first sixteen words of workspace. The timing is given in processor cycles, eg 5Øns on the T424-2Ø , 1ØØns on the T424-1Ø.

occam	**instruction sequence**	**bytes**	**cycles**
x := Ø	load constant Ø	1	1
	store local x	1	1
x := #24	prefix 2	1	1
	load constant 4	1	1
	store local x	1	1
x := y + z	load local y	1	2
	load local z	1	2
	add	1	1
	store local x	1	1

Support for concurrency

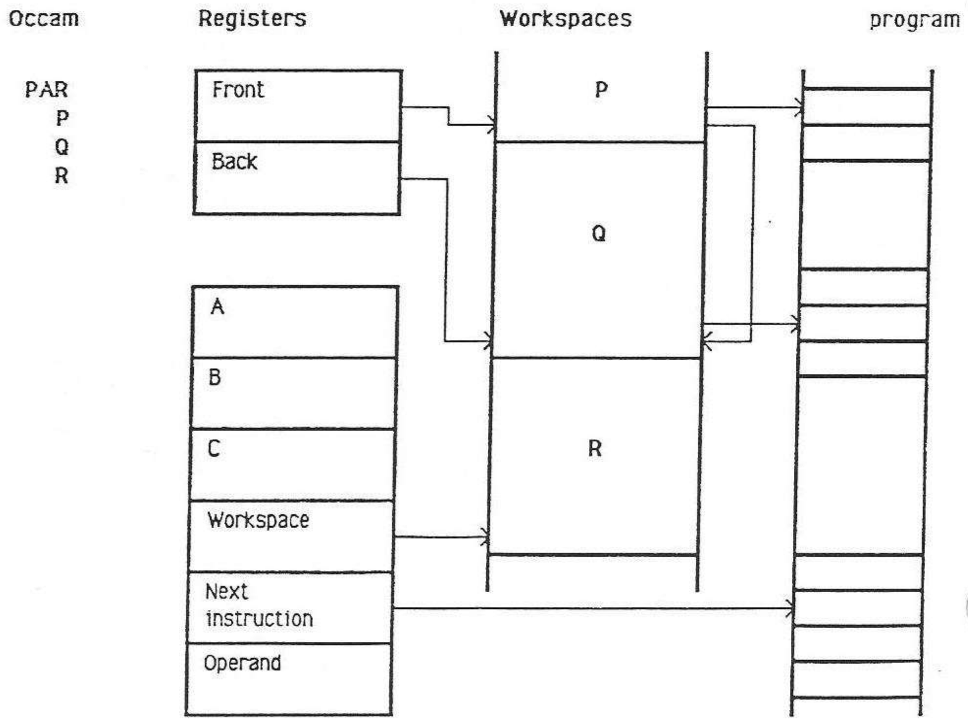
The T424 has a scheduler which enables any number of concurrent processes to be executed together, sharing the processor.

At any time, a process may be

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

The linked list of workspaces of processes is implemented using two registers, **front** points to the first process on the list, **back** to the last. The workspaces contain the variables local to a process, vectors etc, together with some status information.

In this example, process R is executing, and processes P and Q are active, awaiting execution.



Whenever a process is unable to proceed, its instruction pointer is saved in its workspace and the next process is taken from the list. Actual process switch times are very small as little state needs to be saved. The contents of the evaluation stack will not be needed when the process is resumed, and so the evaluation stack is not saved.

Two active lists are maintained, one for each priority.

Communications

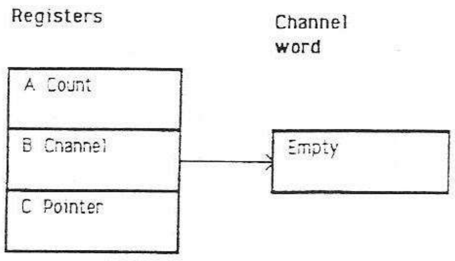
A channel provides a communication path between two processes. Channels between processes executing on the same transputer are implemented by single words in memory (internal channels); channels between processes executing on different transputers are implemented by point-to-point links (external channels).

The processor uses the address of a channel to determine whether the channel is internal or external. This means that the same instruction sequence can be used for both internal and external channels.

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

A process prepares for input or output by loading the evaluation stack with a pointer to a buffer, the identity of the channel, and the count of the number of bytes to be transferred.

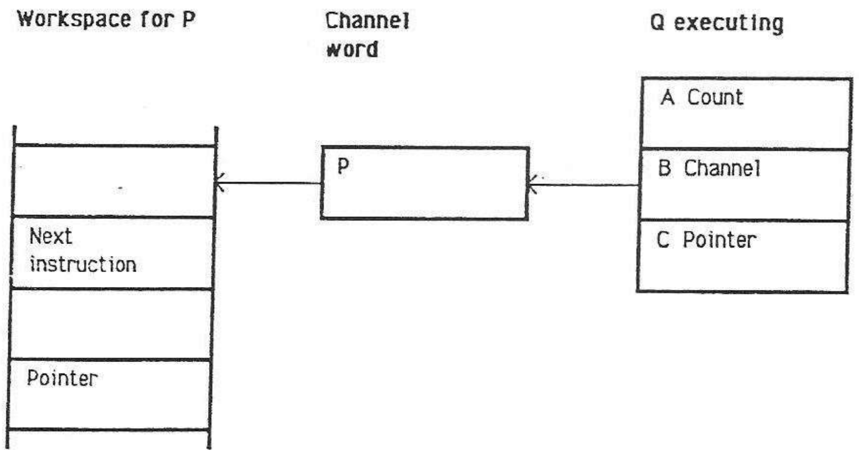
An internal channel is allocated a word in memory, and instructions compiled to initialize it to **empty**.



Internal Communication

When a message is passed using an internal channel, the identity of the first process to become ready is stored in the channel word, and the message pointer stored (with the process context) in the workspace. The processor starts to execute the next process from the scheduling list.

When the second process to use the channel becomes ready, its input/output instruction finds a pointer to the first process in the channel word,



so the message is copied using a block-move, the waiting process is added to the active process list, and the channel reset to the empty state. It does not matter whether the inputting or the outputting process becomes ready first.

External Communication

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

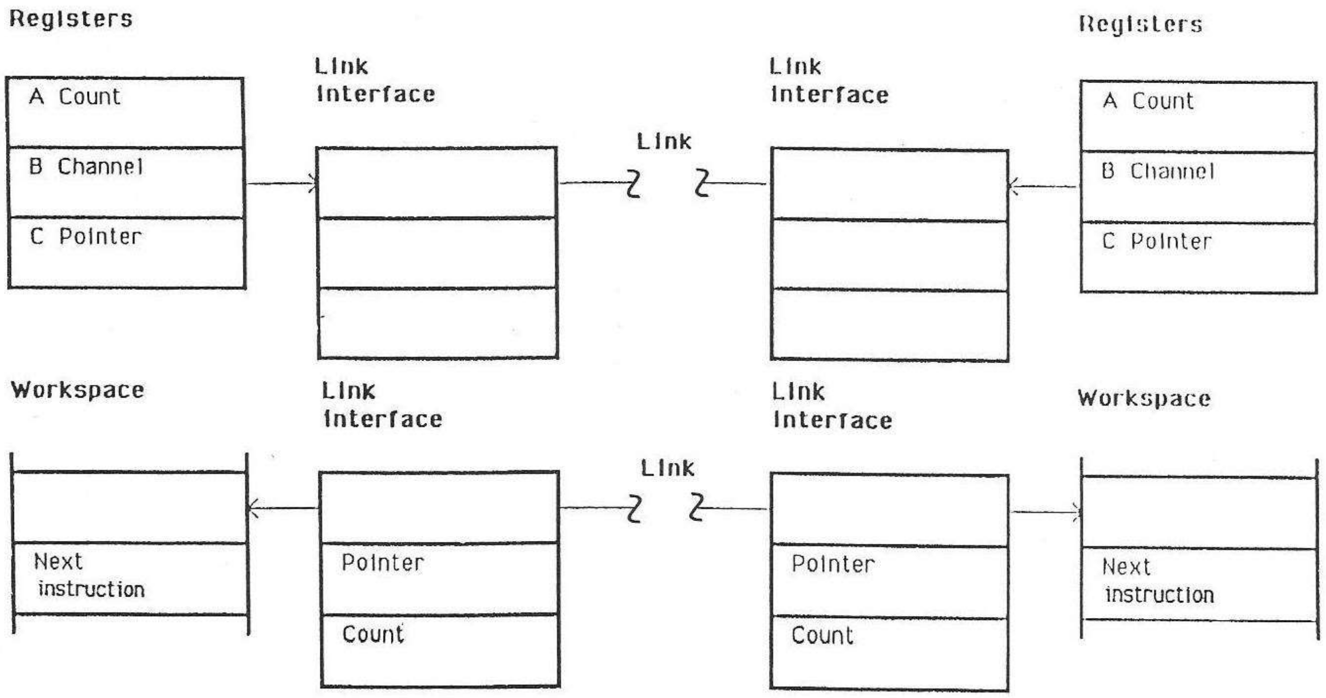
Each process prepares for the transfer as previously described, with count, channel and pointer in the A, B and C registers.

Each of the four link interfaces uses three registers to hold the following information:

a pointer to the workspace of the process
a pointer to the message
a count of bytes to be transferred.

When the 'input message' or 'output message' instruction is executed, these registers are initialized from the processor registers, and the instruction pointer is stored in the process workspace. The processor starts to execute the next process on the scheduling list.

When both processors have initialized both link interfaces, the message is copied, and the link interfaces add the respective processes to the end of the active list.



Detailed descriptions of T424 signals

Signal	Function
ADbus	32-bit external memory address and data bus
	Address. The address is a word address with the bits which would indicate which byte is to be accessed within a word being used for other purposes.
	ADØ is low if one or more byte is to be written
	AD1 is low if the cycle is a refresh cycle
	The address becomes valid at the start of T1 and remains valid until the end of T2.
	Write Data. During a write cycle, data is output on data bytes which are being written; data bytes which are not being written are held high impedence.
	Write data becomes valid (or high impedence) at the start of T3 and remains valid (or high impedence) until the end of T6.
	Read Data. During a read cycle the **AD** outputs are taken high impedence at the start of T3 and remain high impedence until the end of T6.
	Input Data. Input data is sampled at the end of T5, and before any of the control strobes go high at the start of T6. This means that whichever control strobe is used to disable input data, the input data hold time from the strobe is always zero.
	Refresh. During a refresh cycle the **AD** outputs are taken high impedence at the start of T2 and remain high impedence until the end of T6.
AckEvent	Acknowledge external event. See description of **EventIn.**
Analyse	Signal used to investigate the state of a stopped T424. The signal resets the whole of T424 except for the memory interface. On the falling edge of **Analyse** , the processor obeys a bootstrap program which may either be in external ROM or may be received as a message through one of the four INMOS serial links. The program can examine a flag to determine whether the T424 was reset by **Analyse** or **Reset.**
BootFromROM	This signal determines the source of the bootstrap program. If **BootFromROM** is high when **Reset** or **Analyse** goes low, the transputer obeys program from byte location #7FFFFFFE (one byte below the highest byte location in the address space). This byte and the highest byte should contain a jump to the body of the bootstrap program. If **BootFromROM** is low when **Reset** or **Analyse** goes low, the the transputer waits for the first message from one of (any of) the links, and obeys the program received as a message.
BusGranted	See description below of **BusReq** .
BusReq	This input is used by external devices to access the memory interface **ADbus.** **BusReq** is sampled at the end of T6 and in every alternate clock period Tm during which no interface cycle is taking place. If **BusReq** is sampled and found to be high, and if there is no refresh cycle outstanding, the **AD** signals are taken high impedence within two periods Tm of **BusReq** being found to be high. **BusGranted** is taken high one period Tm later. If a refresh cycle is outstanding when **BusReq** is sampled, the refresh cycle is completed before the bus is granted.
	While **BusGranted** is high, **BusReq** is sampled every alternate clock period Tm; if it is found to be low, **BusGranted** is taken low two periods Tm later, and any pending transputer interface cycle can start.
	If the processor and links do not need to access external memory while the bus is granted to a DMA controller, the processor and links can continue accessing internal memory concurrently with the external DMA accesses.

Signal	Function
CapMinus	The more negative terminal of an internal power supply used for the internal clocks. The signal must be connected to the negative terminal of a capacitor of nominal value 1Ø microfarads.
CapPlus	This pin must be connected to the more positive terminal of the 1Ø microfarad capacitor connected to **CapMinus**.
ClockIn	Input clock from which all internal clocks are generated. The nominal clock frequency for all transputers, of whatever wordlength and speed, is 5 MHz.
ErrorOut	The processor has detected an error. All the errors result from overflow, which may be caused by arithmetic overflow, by array bounds violations or by divide by zero.
EventIn	The **EventIn** and **AckEvent** are a pair of handshake signals for external events. External logic takes **EventIn** high when the logic wishes to communicate with a process in the T424. The rising edge of **EventIn** makes a hardware channel ready to communicate with the process. (This channel is a hardware channel inside the T424 additional to the hardware channels of the INMOS serial links.) When both the channel is ready and a process is ready to input from the channel, then the processor takes **AckEvent** high and the process is scheduled. At any time after this point the external logic may take **EventIn** low, following which the processor will set **AckEvent** low. After **AckEvent** goes low, **EventIn** may go high to indicate the next event.
notG	Gate Read Data, Read Enable, Output Enable for external memory. **notG** is taken low during read cycles, including those which occur during configuration, at the start of T4, and is taken high at the start of T6.
GND	Power supply return and logic reference, Ø V. There are several **GND** pins to minimize inductance within the package. The printed circuit inductance should also be minimized.
LinkInØ-3	Input pins of the INMOS serial links. The **LinkIn** pin receives a serial stream of bits including protocol bits and data bits. The quiescent state of the input is low; a Start bit is a high and is followed by a Type bit. Unused link inputs should not be left floating, but should be held to **GND** . If the input is wired to a connector which may have its mating connector connected or disconnected, the input should be held to **GND** via a resistor of less than 1Ø k.
LinkOutØ-3	Output pins of each of the INMOS serial links. The protocol and data bits are as described for the **LinkIn** signals. Link outputs may be left floating or may be connected to one (and only one) **LinkIn** pin. As long as the skew specification is met, the connection may be via buffers.
MConfigC	The **MConfigC** and **MConfigD** inputs are usd to configure the **memory interface. MConfigC** may be connected to GND , VCC , or to any of **ADØ** to **AD15**.
MConfigD	As long as **MConfigC** is connected to one of **ADØ-15**, the **AD** signal to which **MConfigD** is connected determines which of a set of preset configurations is to be used. If **MConfigC** is connected to **VCC** , **MConfigD** is used to input a set of 4Ø one-bit values from an external ROM to the configuration register in response to addresses output by the memory interface. If **MConfigC** is connected to **GND** , the **MConfigD** input is ignored.
ProcClockOut	The processor clock which is output in phase with the memory interface. The processor clock frequency is a multiple of the input clock frequency.

notRefresh Refresh indicator. This signal goes low at the start of T1 if the current cycle is a refresh. It goes high one period Tm after the start of T6.

Reset The falling edge of **Reset** initializes the whole transputer, triggers the sequence which configures the memory interface and then starts the processor obeying a bootstrap program which can either be in external memory or can be received down any one of the four INMOS serial links.

notSØ AddressValid strobe, LatchEnable, or ChipEnable for external memory. The **notSØ** signal goes low at the start of T2 and goes high at the start of T6.

notS1 External memory strobe with configurable width, normally used for RowAddressStrobe, AddressLatchEnable, or ChipEnable. If **notS1** is configured to last for zero periods Tm, it remains high. Otherwise it goes low at the start of T2, and goes high a configurable time (up to 31 periods Tm) later. If the cycle completes before the configured back edge of **notS1** , the strobe is taken high after the end of T6. During a cycle extended by wait states, **notS1** is not extended.

notS2,3,4 Three separate external memory strobes with configurable delay. Their falling edge can be configured to go low between zero and 31 periods Tm after the start of T2, and they go high at the start of T6. If any of these strobes is configured to occur zero periods Tm after the start of T2, it remains permanently low (even during T1 and T6). If it is configured to go low after the end of an interface cycle, it remains high for the complete cycle.

The conventional use suggested for the strobes is:

notS2	use as AddressMultiplex for dynamic RAMs
notS3	use as /CAS for dynamic RAMs
notS4	use as WaitStateGenerator or DataEnable

If the system does not use dynamic RAMs, **notS2** and **notS3** may be used as wait state generators with different delays from **notS4**.

StopProc The rising edge of **StopProc** stops the processor. Memory refresh continues and any outstanding link transfers continue until they need to interact with the processor.

VCC Power supply, nominally 5 V. There are several **VCC** pins to minimize inductance within the package. The printed circuit inductance should also be minimized, and **VCC** should be decoupled to **GND** by at least one 1ØØ nF low inductance (such as ceramic) capacitor.

Wait Wait input for the memory interface. The **Wait** signal is sampled two periods Tm before the end of T4. If, at the time it is sampled, the input is low, the interface cycle proceeds. If, at the time it is sampled, the input is high, the interface is held in T4 until the input is sampled and found to be low; one period Tm later the cycles proceeds to T5.

notWByteØ-3 Separate Write Strobe for each byte of external memory. The **notW** signal for each byte only goes low if the relevant byte is to be written. The write strobes may be configured to be either early or late. Early write strobes allow a wide write pulse to static RAM and allow common I/O for dynamic RAM. Late writes guarantee that data is valid both before and after the write strobe.

In an early write cycle, the **notW** signals are taken low at the start of T3; in a late write cycle at the start of T4. They are taken high at the start of T6.

Memory interface

The memory interface uses a 32-bit wide address/data bus and is configurable on reset to suit a wide variety of different memories and cycle times. One of 9 preset configurations may be used or the configuration may be supplied externally.

The T424 memory cycle has six states, referred to as 'Tstates'. The duration of each Tstate is chosen to suit the memory used.

Tstate

T1	address setup time before address valid strobe
T2	address hold time after address valid strobe
T3	time for the bus to go tristate on a read cycle, or to present valid data on a write cycle
T4,5	time for the read or write data pulse
T6	time for the bus to remain tristate after the end of a read, or for data to remain valid after the end of a write

Each Tstate has a minimum duration of half the processor cycle time, and (except for T4) a maximum duration of two processor cycle times. T4 may be extended by wait states. The unit of half the processor cycle time is referred to as the interface clock period, Tm.

Control Signals

Separate write strobes **notWbyteØ** , **notWbyte1** , **notWbyte2**, **notWbyte3** , are provided for each byte. The read strobe, **notG** , ensures the bus is tristated before read data is enabled.

Five user-definable strobes, **notSØ** to **notS4** are provided. **notS1** to **notS4** can be configured independently. They may be used directly to control multiplexed-address dynamic RAMs.

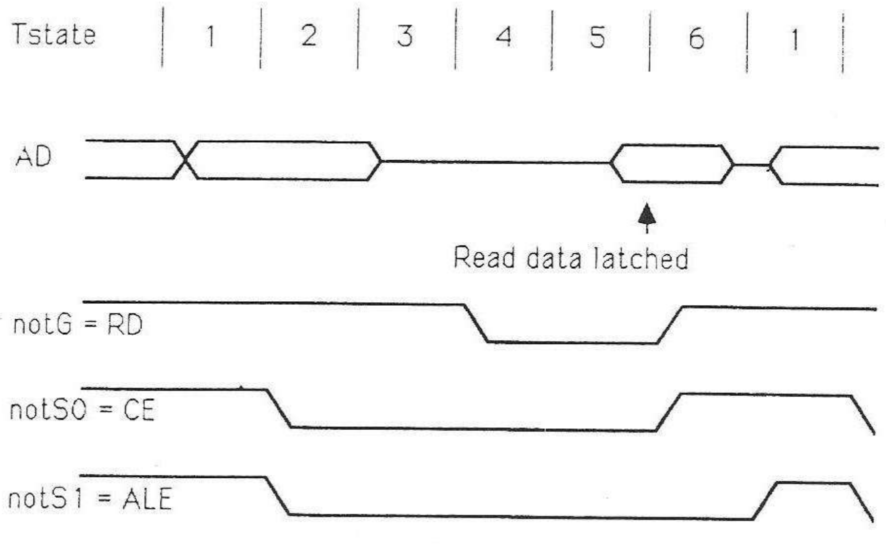
notSØ starts at the start of T2 and ends at the start of T6.

notS1 starts at the start of T2, ends up to 31 periods later.

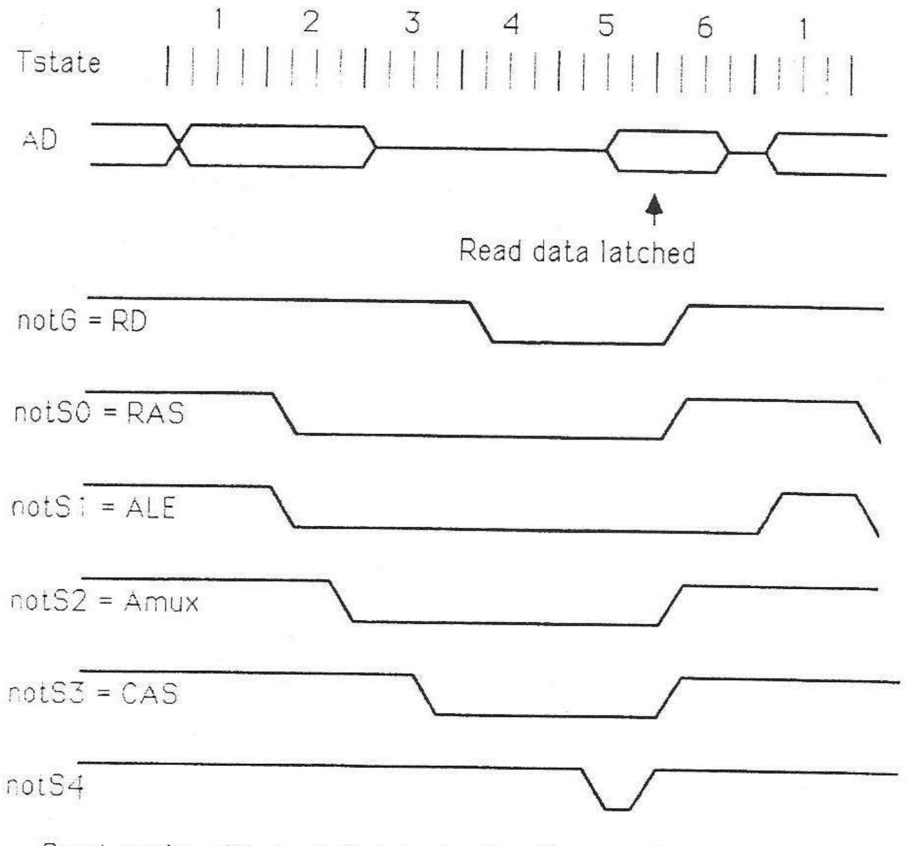
notS2, notS3, notS4 start up to 31 periods after T2 and end with the start of T6.

Read

Read cycles generate **notSØ-notS4 and the notG** gate data signal. Input data is latched into the memory interface at the end of T5. Byte reads are performed as word reads, with byte selection performed inside the transputer. The byte address bits are not output on the AD signals.



Read cycle with each Tstate lasting one period Tm



Read cycle with each Tstate lasting four periods Tm

Write

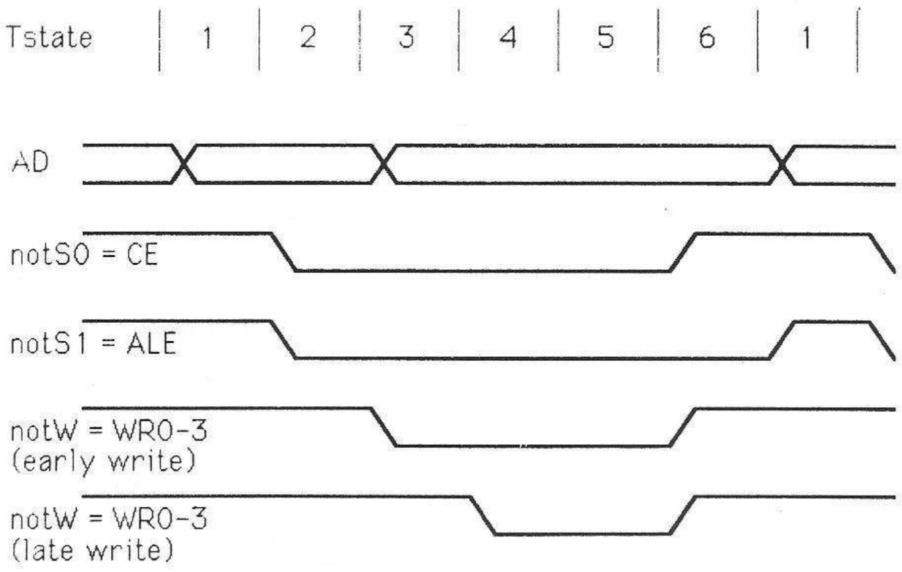
Write cycles generate **notSØ-notS4** and the **notW** write data strobe signals. Separate **notW** signals are provided for each byte. A word write uses all the **notW** signals; if a particular byte is not to be written the relevant **notW** signal is not generated, and the corresponding data outputs are tristated.

Write cycles may be configured to be either early or late. In late write cycles, write data is valid before the leading edge of **notW** and after the back edge of **notW** .

Early write cycles provide a wide **notW** pulse, with data valid both before and after the back edge of **notW.**

The **notG** signal is held high during write cycles

Early indication of a write cycle is provided by a low siignal on **ADØ** while the address is valid.



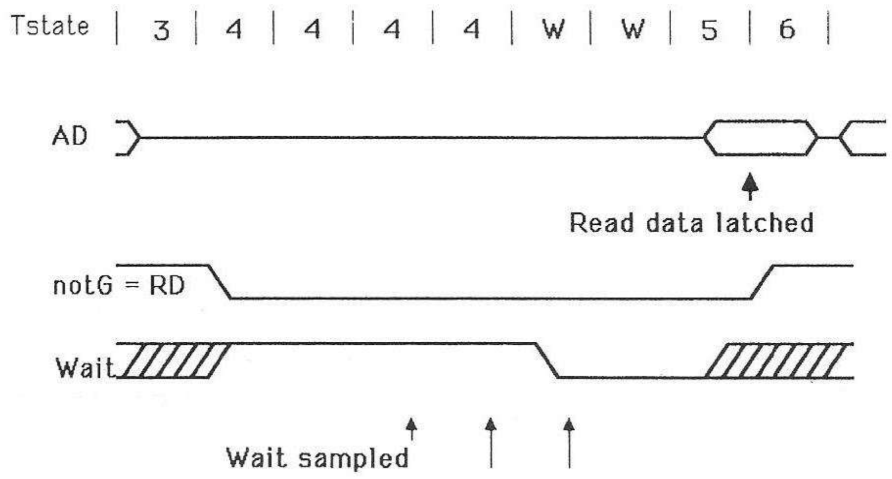
Write cycles -- early and late

Wait States

Wait states generated by the configurable strobes can extend the interface cycle up to over 18 processor cycles for slow external devices - over 900ns for T424-20.

An interface cycle may be extended by holding the **Wait** input high. This may be done using one of the configurable strobes, **notS1 - notS4** , or an external signal. The **Wait** input is sampled one period Tm before the end of T4 and thereafter every clock period until it is detected low, when the cycle continues with T5.

Note that if **Wait** is used to extend cycles for longer than the interval between refresh cycles, refresh cycles are liable to be lost.



Cycle configured for long T4, with **Wait** extending the cycle by two periods.

Refresh

Refresh can be configured to be enabled or disabled. If Refresh is enabled, the interval between refresh cycles can be configured to be a number of periods of the input clock signal **ClockIn** :

Refresh interval (periods of ClockIn)	Interval with ClockIn frequ-ency = 5.0 MHz (microseconds)
72	14.4
54	10.8
36	7.2
18	3.6

Configurable refresh intervals.

Refresh cycles generate all the strobes **notS1** to **notS4** , but neither **notG** nor the **notW** signals. **notRefresh** goes low during T1 and remains low until the end of T6. Refresh is also indicated on AD1 during T1 and T2, with the same timing as address signals.

A refresh cycle outputs a 9 bit refresh address, on **AD2-10** . It is decremented after each refresh cycle. The remaining address bits (including address bit 31) are high for a refresh cycle.

Memory Interface Initialisation

When **Reset** is high, the whole T424 is held in a reset state. When **Reset** goes low, the memory controller goes through an initialization sequence, and selects the memory interface configuration setting. The interface controller then lets the rest of the transputer start initialization.

During initialization and configuration, the memory interface controller performs some memory cycles. It also carries out eight refresh cycles, so that external dynamic RAM can be initialized. The BusReq signal is ignored until the configuration is complete.

The initialization and configuration sequence is shown below. The complete sequence is always performed, but if the configuration disables refresh, the refresh cycles are dummy cycles which do not generate the control signals **notG**, **notW** , or **notRefresh** .

The sequence is:

- The **Reset** signal goes low.
- Delay of 144 periods of the input clock,ClockIn.
- The **ADbus** signals are scanned to see which (if any) of the preprogrammed interface configurations is selected. The scan lasts for 144 clock periods.
- The memory interface performs 4Ø read cycles at the slowest interface speed to access the configuration (if any) in external ROM.
- Delay dependent on the selected configuration.
- Eight refresh cycles to initialize dynamic RAM.
- The rest of the T424 is allowed to start.

Interface configuration details

Either the preset internal settings or settings held in external memory (usually ROM) can be used. Two signals -- **MConfigc** and **MConfigD** control this.

The **MConfigC** input determines how the **MConfigD** input is used. **MConfigC** may be connected to GND, VCC, or to any of **ADØ** to **AD15** . These connections have effect as follows:

MConfigC connected to	Effect
GND	Use default setting, with refresh enabled and interface cycle set to slowest. Ignore the **MConfigD** input.
VCC	Use none of the preprogrammed settings, but configure from external ROM via the **MConfigD** input.
ADØ-7	Use the preprogrammed setting defined by which of the **AD** pins **MConfigD** is connected to. (The particular **ADØ-7** pin that **MConfigC** is connected to is irrelevant, but it is usually simplest to wire **MConfigC** to the same pin as **MConfigD**).

The table below gives the preprogrammed configurations. The numbers underneath each Tstate give the number of periods Tm in that Tstate. The numbers under **notS1** give the pulse width of **notS1** in periods Tm. The numbers under **notS2** to **notS4** give the delay from the start of T2 to the falling edge of each strobe, in periods Tm. The number under Refresh gives the number of periods of the input clock between refresh cycles, e.g. 72 periods of 200 ns give a refresh interval of 14.4 microseconds. The pin on the left hand side of the table is the **ADbus** signal which should be connected to **MConfigC** and **MConfigD** to select that configuration setting.

	Duration of Tstate						Posn. of movable edge not				Type of write cycle	Refresh Interval (Input clocks)
pin	T1	T2	T3	T4	T5	T6	$\overline{S1}$	$\overline{S2}$	$\overline{S3}$	$\overline{S4}$		
AD0	3	3	3	3	3	3	30	2	4	13	late	72
AD1	2	2	2	2	2	2	30	2	3	9	late	72
AD2	2	3	1	1	2	3	30	2	3	9	late	72
AD3	1	2	1	2	2	2	7	1	2	3	early	72
AD4	1	2	1	1	1	2	30	1	2	3	late	72
AD5	1	1	1	1	1	1	30	2	7	17	early	-
AD6	1	1	1	2	2	1	5	1	2	9	early	72
AD7	1	1	1	1	1	1	3	1	2	3	early	72
AD8	1	1	1	1	1	1	3	1	2	3	late	72

Preprogrammed memory interface configurations.

pin	Comment	notS0	notS1	notS2	notS3	notS4
AD0	gen. purpose slow	CE or RAS	ALE	Amux	CAS	Wait
AD1	general purpose	CE or RAS	ALE	Amux	CAS	Wait
AD2	slow 2630, fast T424	CE or RAS	ALE	Amux	CAS	Wait
AD3	2620-10, possibly-12		RAS	Amux	CAS	Data
AD4	fast 2630 or 2830	CE or RAS	ALE	Amux	CAS	Data
AD5	1400-70, lots of waits	CE	-----WaitGenerators----			
AD6	2800-10 in 200 ns		RAS	Amux	CAS	Wait
AD7	superfast dynamic					
AD8	superfast dynamic					

Preprogrammed configurations -- signal functions

To determine which configuration setting to use, the particular setup times, hold times, access times, and cycle times that the memory system needs must be calculated. A program is provided with the Transputer Development System for working out the actual interface timings for a particular configuration.

For other timing configurations, **MConfigC** is connected to VCC, and **MConfigD** is used to input a set of 40 one-bit values from an external ROM. For the full coding details of the external ROM, see the 'T424 Reference Manual'.

INMOS serial links

The IMS T424 has four standard INMOS serial links for high speed intercommunication between transputer products. Each link operates independently, and provides block transfers for messages into and out of the T424. There are eight DMA channels,two per link,which can access the whole of the T424's address space.

Each link implements two occam channels. The link interface consists of an output and an input signal, both of which are used to carry data and link control information. A message is transmitted as a sequence of bytes. After transmitting a data byte, the sending transputer waits until an acknowledge has been received, signifying that the receiving transputer is ready to receive another byte. The receiving transputer can transmit an acknowledge as soon as it starts to receive a data byte so that transmission is normally continuous. This asynchronous protocol guarantees reliable transmission in spite of possible delays in either the sending or the receiving transputer.

The links support a communications frequency of twice the input clock frequency -- i.e. 10 Mbit/s with a 5 MHz input clock.

An internal link clock is derived from the transputer's input clock and data bits are transmitted synchronously with this clock. Data reception is asynchronous.

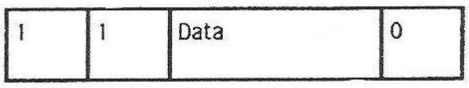
Links can be used with the INMOS Link Adaptor to provide a set of handshaken byte-wide ports.

Link protocol

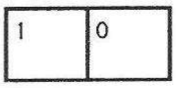
Each signal line carries data packets and control packets. A message is transmitted as a sequence of packets. Each packet contains a logic 'one' start bit followed by a control bit. In data packets, this is followed by the eight data bits and a stop bit. After transmitting a data packet, the sender waits until an acknowledge packet is received.

For data packets, the control bit is set to 'one', for acknowledge packets to 'zero'.

Data packet



Acknowledge packet



The quiescent state of the link signals is logic zero.

Data and acknowledge packets are interleaved on each signal line. An acknowledge packet is transmitted as soon as the data packet has been identified, provided that there is sufficient buffer space for another data packet and the inputting process is ready for the communication. Thus the flow of data packets can be continuous.

Connecting transputers with INMOS serial links

The INMOS serial links are TTL level signals intended primarily for communication between devices on the same printed circuit board or between printed circuit boards via a backplane. The link signals require faithful transmission of pulse widths, so the propagation delay of the rising edge must be the same, within a small tolerance, as the propagation delay of the falling edge.

The timing specification for the links is such that the universal link speed can be buffered, so long as care is taken over the skew between TPLH and TPHL.

Buffering

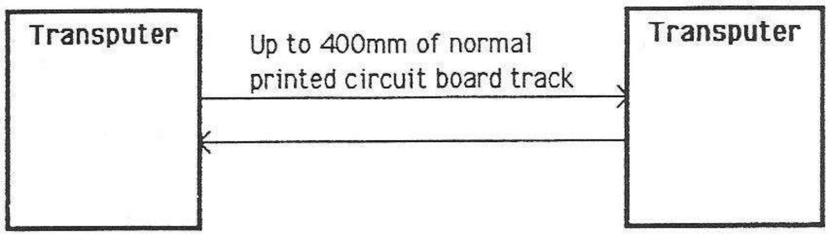
Distances up to 4ØØmm can be connected by transputers using unbuffered links.

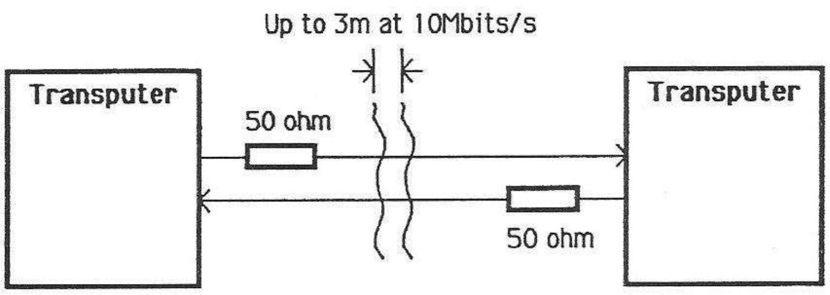
Series termination resistors of the order of 5Ø ohms can be used for extending this to 3 metres.

The important parameter for any link buffers is the difference between TPHL and TPLH, which is not specified for most families of TTL. The difference between the propagation delays causes widening or narrowing of the data pulses; if the pulse width change does not meet the specification the data will be lost. (see Link AC characteristics)

Using RS 422 drivers and receivers (e.g. 26LS31, 26LS32) up to 12 metres of distance can be used.

Two transputers running at different frequencies can be connected via a pair of INMOS Link Adaptors.





T424 Performance

Performance is measured in terms of the number of bytes to hold the program, and the number of (internal) processor cycles to execute it. A processor cycle is 5Øns on a T424-2Ø , 1ØØns on a T424-1Ø.

For comparable constructs, an efficient implementation of other languages should achieve approximately the same performance as occam.

A performance estimator program is one of the T424 design tools available. Using simple static analysis of the given occam program, it is able to provide a static performance analysis, related to the program's hierarchical process structure, which is accurate to within 1Ø%. The tables given below allow a preliminary, manual assessment to be made.

The code size of a program is estimated by summing the size for each of its elements. The execution time may similarly be estimated.

For example, using the table given at the end of this section, and assuming that **a** , **b** and **c** have been declared as **INT** or **INT32** ,

a := b / c

requires, on average, 1.1 bytes of code for each variable, two bytes for the division, and none for the assignment, totalling 5.3 bytes. To execute the process requires, on average, 2.1 processor cycles each to read b and c , 4Ø processor cycles for the division, and 1.1 processor cycles to store the result in **a**, totalling 45.3 cycles (an average of 2.3 microseconds on the T424-2Ø).

Floating point operations

Floating point operations are provided by a run-time package, which requires approximately 2ØØØ bytes of memory. Ancilliary operations (eg type conversion) require an additional 2ØØØ bytes approximately. The following table summarizes the performance of the package

		Processor cycles		
		best	typical	worst
REAL32	+, -	3ØØ	36Ø	54Ø
	*,/,\\	34Ø	34Ø	4ØØ
	<,>,=,>=,<=,<>	6Ø	6Ø	6Ø
REAL64	+, -	34Ø	4ØØ	58Ø
	*,/,\\	38Ø	38Ø	44Ø
	<,>,=,>=,<=,<>	6Ø	6Ø	6Ø

Performance summary

The following abbreviations are used to represent the quantities indicated in the tables below:-

k:	lexical difference between declaration and use
np:	number of component processes
r:	1 if INT or vector parameter, Ø if not
ts:	number of table entries (table size)
w:	width of constant in nibbles
p:	number of places to shift
Eg:	expression used in a guard
Et:	timer expression used in a guard

	size bytes	time cycles
Constants		
load constant	w	w
Identifiers		
variables in expression	1.1+k+r	2.1+2(k+r)
assigned to or input to	1.1+k+r	1.1+(k+r)
in PROC call, corresponding to an INT parameter	1.1+k+r	1.1+(k+r)
channels	1.1+k	2.1+2k
Vector Variables		
constant subscript	Ø	Ø
variable subscript	5.3	7.3
expression subscript	5.3	7.3
byte access overhead	Ø	+2
Table access		
variable subscript	9.3	13.3
expression subscript	9.3	13.3
i.e. table overhead	4	6
Declarations		
INT workspace adjustment if new process		
CHAN	3.1	3.1
CHAN[size]	9.4	2.2 + 2Ø.2*size
TABLE	ts*4	Ø
byte TABLE	ts	Ø
PROC	body+2	Ø
DEF	Ø	Ø
Primitives		
assignment	Ø	Ø
input	4	26.5
output	1	26
stop	2	25
SKIP	Ø	Ø

Arithmetic operators		
add	1	1
subtract	1	1
multiply	2	39
divide	2	4Ø
remainder	2	38
rshift	2	3+p
lshift	2	3+p
Comparison operators		
= constant	Ø	1
= variable	2	3
<> constant	1	3
<> variable	3	5
>	1	2
<	1	2
>=	2	4
<=	2	4
Logical operators		
OR	4	8
AND	1	2
NOT	1	2
/\	2	2
\/	2	2
><	2	2
not	2	2
Expressions		
check if error	4	6
Constructs		
SEQ	Ø	Ø
IF	1.3	1.4
if guard	3	4.3
ALT (non timer)	8	31
ALT (timer)	8	71
alt channel guard	8+2(k+1.1)+2Eg	17.5+2(k+1.1)+2Eg
timer alt guard	8+2Eg+2Et	41+2Eg+2Et
skip alt guard	8+2Eg	1Ø+2Eg
PAR	11.5+(np-1)*	19.5+(np-1)*3Ø.5
WHILE	4	12
Procedure call	3.5+(nparams-2)*1.1 +nvecparams	16.5+(nparams-2)*1.1 +nvecparams*2.3
Replicators {figures in curly brackets are not necessary if the number of replications is a compile time constant}		
replicated SEQ	7.3{+5.1}	-3.8+15.1*count{+7.1}
replicated IF	12.3{+5.1}	-2.6+19.4*count{+7.1}
replicated ALT	24.8{+1Ø.2}	25.4+33.4*count{+14.2}
replicated time ALT	24.8{+1Ø.2}	62.4+33.4*count{+14.2}
replicated PAR	39.1{+5.1}	-6.4+7Ø.9*count{+7.1}

Effect of off-chip memory

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

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

External memory is characterized by the number of extra processor cycles per external memory cycle, denoted as **e** . **e** has value 2 for the fastest external memory, a typical value for a large off-chip memory is 5.

In general, the number of additional cycles required to access a variable which is stored in off chip memory is **e** . If the program is stored off chip, and **e** has the value 2 or 3, then, in general, no extra cycles are required for linear code sequences. For larger values of **e** , the number of extra cycles required for linear code sequences may be estimated at **(e - 3) / 4** . A transfer of control requires **e + 3** cycles.

These rules of thumb may be refined for various constructs, as given in the following table which indicates the number of additional cycles which result from the use of external memory. **n** denotes the number of components in a construct. In the case of **IF** , the **n** 'th conditional is the first to evaluate to **TRUE** , and the costs include the costs of the conditionals tested. **b** denotes the number of bytes in a slice assignment or communication.

	program off chip	data off chip
Boolean expressions	e - 2	Ø
IF	3en - 8	en
Replicated IF	(6e - 4)n + 7	(5e - 2)n + 8
Replicated SEQ	(3e - 3)n + 2	(4e - 2)n
PAR	(3e - 1)n + 8	3en + 4
Replicated PAR	(1Øe - 8)n + 8	16en - 12
ALT	(2e - 4)n + 6e	(2e - 2)n + 1Øe - 8
slice assignment and communication within the same T424	Ø	max (2e, e(b/2))

Absolute maximum ratings (see note 1)

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

Notes:

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

2 All voltages are with respect to GND.

3 This device contains circuitry to protect the inputs against damage caused by high static voltages or electric fields; however it is advised that normal precautions be taken to avoid application of any voltage higher than the maximum rated voltages to this high impedence circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level such as GND.

Recommended operating conditions

Parameter		Min	Max	Unit	Note
VCC	DC supply voltage	4.5	5.5	V	1
VI,VO	Input or output voltage	0	VCC	V	1,2
II	Input current		±25	mA	3
CL	Load capacitance on any pin			pF	4
CLbus	Total load capacitance on ADbus			pF	4
TA	Operating temperature range	0	70	°C	

Notes:

1 All voltages are with respect to GND.

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

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

4 These capacitances will be specified on the parameter list as a result of full characterization.

DC characteristics

$4.5\ V \leq VCC \leq 5.5\ V$, $0°C \leq TA \leq 70°C$, input clock frequency = 5 MHz
All voltages are with respect to GND

Parameter		Conditions	Min	Max	Unit	Note
VIH	High level input voltage		2.0	VCC+0.5	V	
VIL	Low level input voltage		*	0.8	V	1
II	Input current	GND ≤ VIN ≤ VCC		±200	μA	
VOH	Output high voltage	IOH = -2mA	VCC-1		V	
VOL	Output low voltage	IOL = 4mA		0.4	V	
IOS	Output short circuit current	GND ≤ VOUT ≤ VCC	*	*		1
IOZ	Output current in tristate	GND ≤ VOUT ≤ VCC		±200	μA	
PD	Power dissipation			*	W	1
CIN	Input capacitance	f = 1 MHz		*	pF	1
COZ	Output capacitance in tristate	f = 1 MHz		*	pF	1

Notes:
1 Entries marked with an asterisk (*) will be specified on the parameter list as a result of full characterization.

AC characteristics of INMOS serial links

Parameter	Max	Unit	Note
Difference in frequency of input clock ClockIn between two devices connected by a link	400	ppm	1
Skew in buffering between delay to rising edge and delay to falling edge		ns	2
Synchronization failure rate	0.1	FIT	3

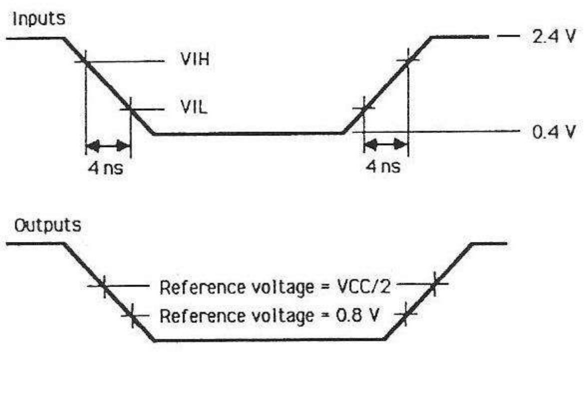
Notes:
1 The difference in frequency of 400 ppm allows the use of low cost ±200 ppm crystal oscillators.
2 The absolute delay in buffering between a LinkOut pin and a LinkIn pin is immaterial because data reception is asynchronous. Waveform fidelity is important, however, and this is most easily expressed as a skew tolerance. The effect of skew is to narrow the pulse widths. The actual permissible skew will be specified on the parameter list as a result of full characterization.
3 A FIT is one failure in one thousand million device operating hours. Synchronization failure is fundamental to any system which samples data asynchronously. It is possible to reduce the probability of synchronization failure to any desired small probability. The figure specified has been chosen to be substantially lower than the failure rate from reliability failures. The figure is guaranteed by design and characterization and is not tested.

AC characteristics of memory interface

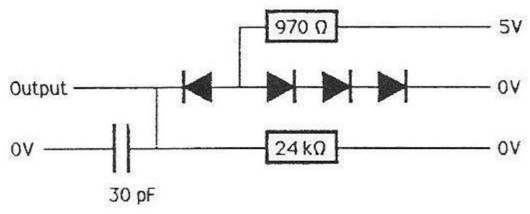
IMS T424	speed selection	T424-10		T424-12		T424-15		T424-17		T424-20	
Tm	Interface clock period	50		40		33		28		25	
		min	max	min	max	min	max	min	max	min	max
TOLOL	Cycle time	295		235		193		163		145	
TAVQV	Address access time		200								
TOLQV	Access time from notS0		160								
TGLQV	Access time from notG		60								
TAVOL	Address set-up time										
TOLAX	Address hold time										
TGHQX	Read data hold time	0									
TGHQZ	Read data turn-off										
TOLOH	notS0 pulse width low	180									
TOHOL	notS0 pulse width high	84									
TOLWL	notS0 to notW delay										
TDVWL	Wr data set-up to notW										
TDV3L	Wr data set-up to notS3										
TWLDX	Wr data hold time 1										
TWHDX	Wr data hold time 2										
TWLWH	Write pulse width										
TWLOH	Effective notW width										
T1L1H	notS1 pulse width low										
T1H1L	notS1 precharge time										
T3L3H	notS3 pulse width low										
T3H3L	notS3 precharge time										
T1L2L	notS1 to notS2 delay										
T2L3L	notS2 to notS3 delay										
T1L3L	notS1 to notS3 delay										
T1LQV	Access time from notS1										
T2LQV	Access time from notS2										
T3LQV	Access time from notS3										
T3HQZ	Read data turn-off										
T3L1H	notS1 hold from notS3										
T1L3H	notS3 hold from notS1										
T1LWH	notW hold from notS1										
T1LDX	Wr data hold from notS1										
TWL3H	notW to notS3 lead time										
TWL1H	notW to notS1 lead time										
T4LYL	Gate delay to Wait input										
TRFSH	256 refresh cycles (µs)		3650								

All parameters listed are for example purposes only. Parameters will be fully specified on the parameter list as a result of full characterization.

Reference points for AC characteristics



Load circuit



AC characteristics of system services and processor signals

Parameter		Min	Max	Unit	Note
TCHCL	Clock pulse width high	40		ns	
TCLCH	Clock pulse width low	40		ns	
TR	Clock rise time from VIL to VIH	1.5	10	ns	
TF	Clock fall time from VIH to VIL	1.5	10	ns	1
TCLCL	Clock period	200	400	ns	2
ΔTCLCL	Instability of clock period		±250	ps	3
Cap	Electrolytic decoupling capacitor between CapPlus and CapMinus	8	20	μF	4
TRHRH	Reset pulse width high	8		ClockIn periods	
	Time VCC must be valid and ClockIn running before Reset goes low	10		ms	
	BootFromROM setup time before Reset or Analyse goes low	0			
	BootFromROM hold time after Reset goes low	50		ms	
	ErrorOut pulse width	2		processor cycles	
	Stop pulse width	2		processor cycles	
	Analyse pulse width	8		ClockIn periods	

Notes:

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

2 The TCLCL parameter must be met at all voltages between VIH and VIL.

3 The ΔTCLCL parameter is required to be met by any T424 using any of its INMOS serial links. The tolerance of ±250 ps allows instability of the electrical signal of 1250 ppm at 5 MHz, which compares with the 200 ppm recommended tolerance for the crystal of a crystal oscillator.

4 This specification is met by a 10 μF, +80% -20%, 10 V capacitor.

AC characteristics of peripheral interfacing

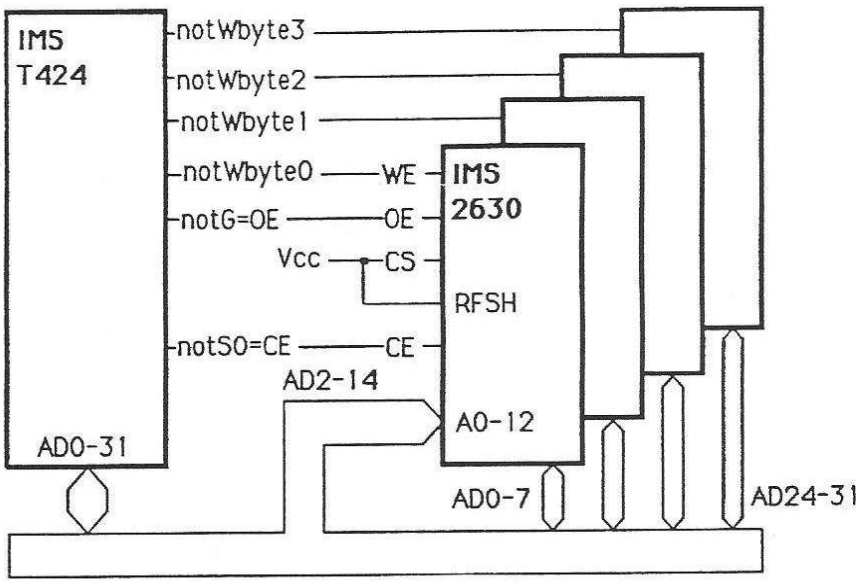
Parameter		Min	Max	Unit	Note
TVHVL	EventIn pulse width high	2		processor cycles	
TVLVH	EventIn pulse width low	2		processor cycles	
TVHKH	Delay from EventIn to AckEvent	0	2	processor cycles	1
TVLKL	Falling edge delay from EventIn to AckEvent	0	2	processor cycles	
TKHKL	AckEvent pulse width high	2		processor cycles	
TKLKH	AckEvent pulse width low	2		processor cycles	
TKHVL	Delay from AckEvent to falling edge of EventIn	0			
TKLVH	Delay from falling edge of AckEvent to next EventIn	0			
	BusReq to BusGranted delay	0	2 +2	memory cycles processor cycles	2
	ADbus tristate before BusGranted	0			
	ADbus tristate before BusReq goes low	0			
	Delay from BusReq going low to BusGranted going low	0			

Notes:

1 The maximum delay is as specified if a process has already input from the Event channel. If no process has input from the Event channel the AckEvent signal does not go high until a process does input from the channel. If no process inputs from the channel the Event is not acknowledged.

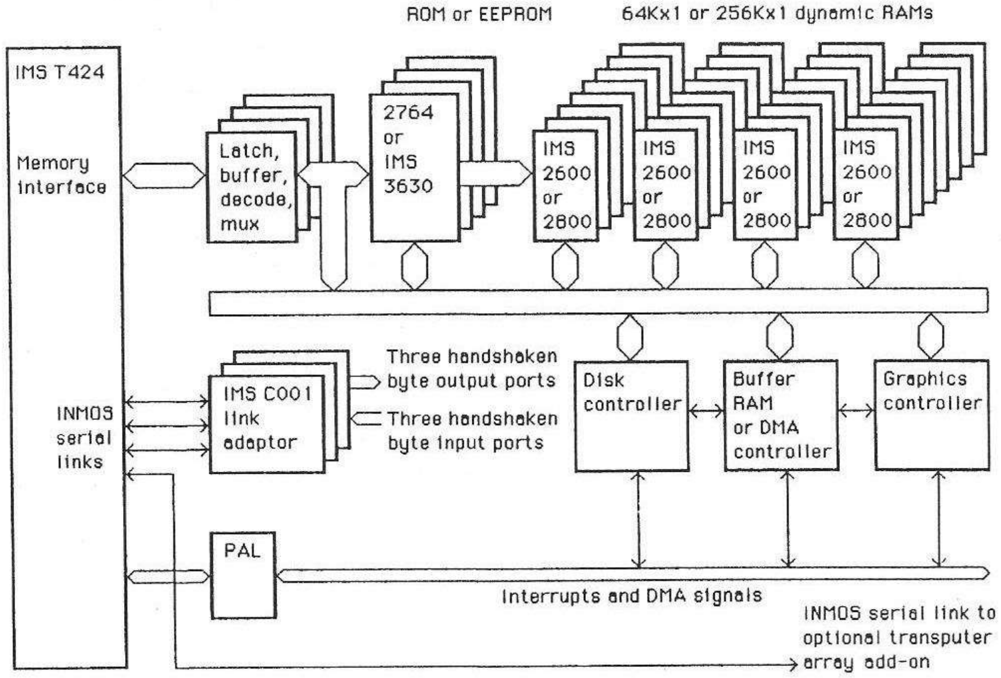
2 BusReq is sampled at the end of each memory cycle. If a refresh cycle is outstanding, the refresh cycle is completed before the bus is granted.

Small single transputer system

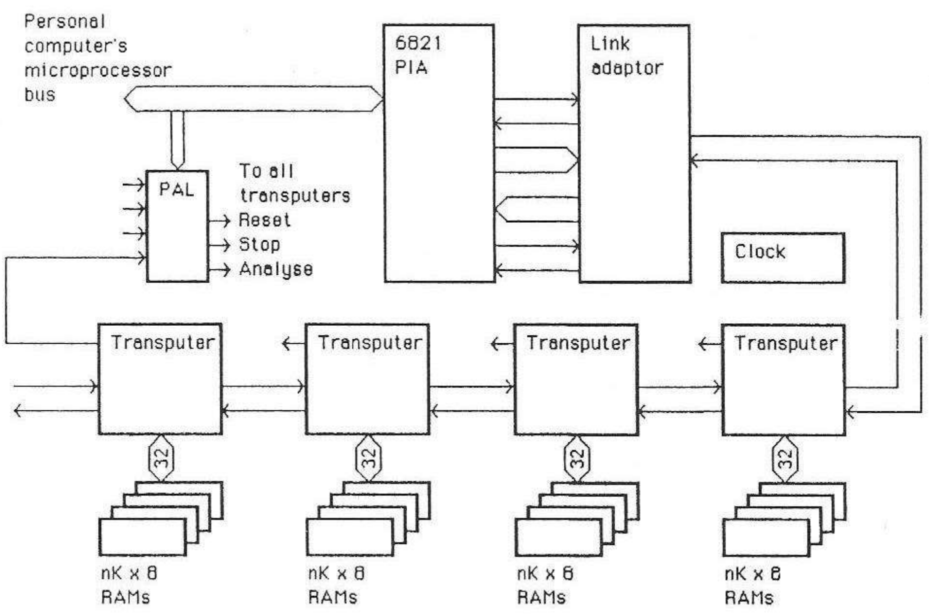


Large single transputer system

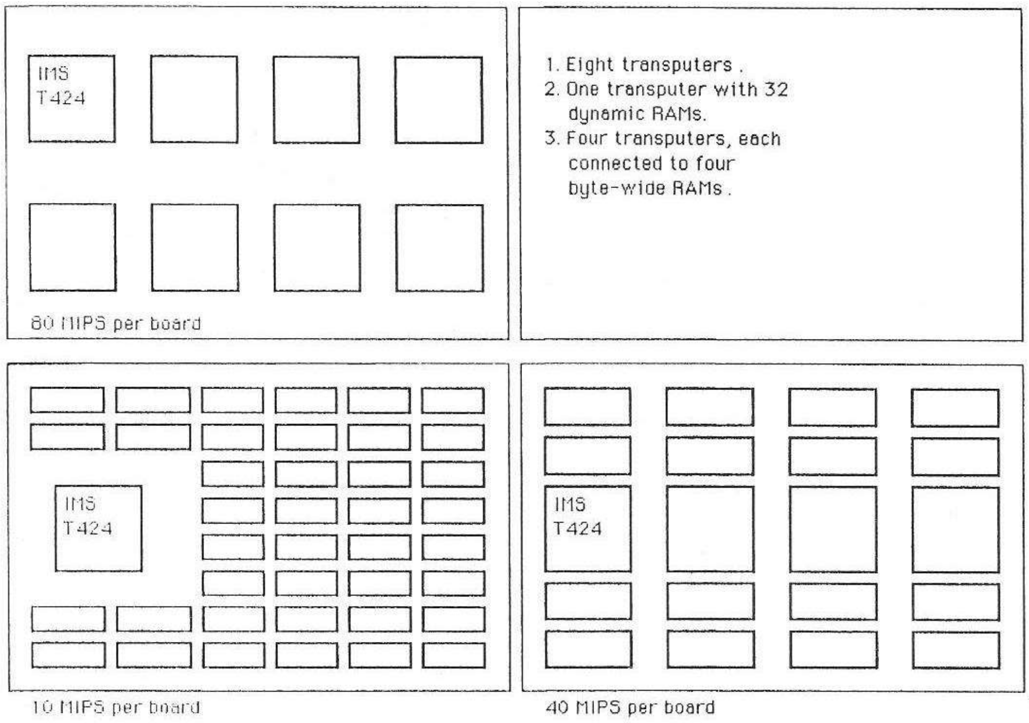
High performance workstation


ROM or EEPROM
64Kx1 or 256Kx1 dynamic RAMs
IMS T424
Memory interface
Latch, buffer, decode, mux
2764 or IMS 3630
IMS 2600 or 2800
IMS 2600 or 2800
IMS 2600 or 2800
IMS 2600 or 2800
INMOS serial links
IMS C001 link adaptor
Three handshaken byte output ports
Three handshaken byte input ports
Disk controller
Buffer RAM or DMA controller
Graphics controller
PAL
Interrupts and DMA signals
INMOS serial link to optional transputer array add-on

Add-on board for personal computer



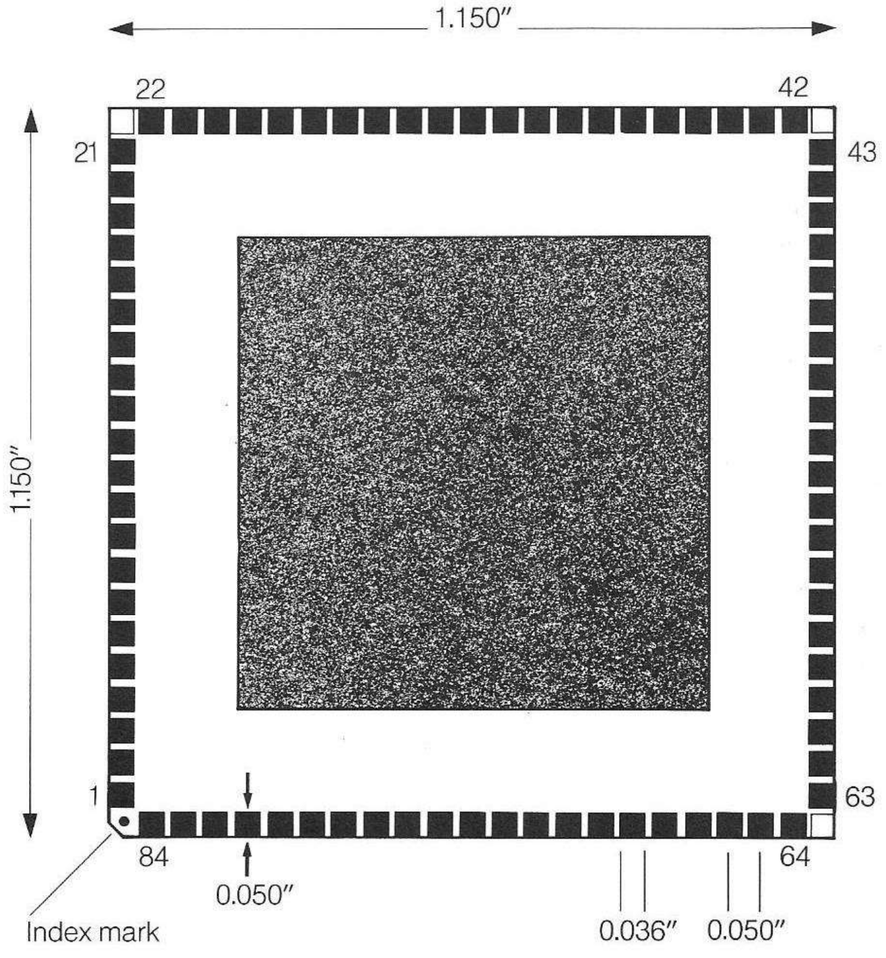
Examples of double Eurocards



IMS T424
80 MIPS per board
1. Eight transputers .
2. One transputer with 32 dynamic RAMs.
3. Four transputers, each connected to four byte-wide RAMs .
IMS T424
IMS T424
10 MIPS per board
40 MIPS per board

Notes

Packaging



JEDEC type A 84 contact leadless chip carrier

T424 is designed to provide a number of speed selections:

Designation	Instruction throughput	Processor clock speed	Processor cycle time	Input clock frequency
IMS T424-1Ø	5 MIPS	1Ø MHz	1ØØ ns	5 MHz
IMS T424-12	6 MIPS	12.5 MHz	8Ø ns	5 MHz
IMS T424-15	7.5 MIPS	15 MHz	67 ns	5 MHz
IMS T424-17	9 MIPS	17.5 MHz	57 ns	5 MHz
IMS T424-2Ø	1Ø MIPS	2Ø MHz	5Ø ns	5 MHz

<u>**Associated Documentation**</u>

Transputer Reference Manual,	INMOS	1984
Occam Programming Manual	INMOS	1982 et seq
	also Prentice Hall	1984 et seq
Transputer Development System Manual in preparation		

inmos

INMOS Limited
Whitefriars
Lewins Mead
Bristol BS1 2NP
UK
Telephone (0272) 290861
Telex 444723

INMOS GmbH
Danziger Strasse 2
8057 Eching
West Germany
Telephone (089) 3191028
Telex 522645

INMOS SARL
Immeuble Monaco
7 rue le Corbusier
SILIC 219
94518 Rungis Cedex
France
Telephone (1) 687 22 01
Telex 201222

INMOS Corporation
PO Box 16000
Colorado Springs
Colorado
CO 80935
USA
Telephone (303) 630 4000
Telex 910 920 4904

August 1984

72-TRN-007-000