

inmos

TRANSPUTER DEVELOPMENT AND *iq* SYSTEMS DATABOOK

Second Edition 1991

**SGS-THOMSON**
MICROELECTRONICS

INMOS is a member of the SGS-THOMSON Microelectronics Group

INMOS Databook series

Transputer Databook

Military and Space Transputer Databook

Transputer Development and *iq* Systems Databook


Graphics Databook

Transputer Applications Notebook: Architecture and Software

Transputer Applications Notebook: Systems and Performance

Copyright © INMOS Limited 1991

INMOS reserves the right to make changes in specifications at any time and without notice. The information furnished by INMOS in this publication is believed to be accurate; however, no responsibility is assumed for its use, nor for any infringement of patents or other rights of third parties resulting from its use. No licence is granted under any patents, trademarks or other rights of INMOS.

 **mos**[®], IMS and occam are trademarks of INMOS Limited.

INMOS is a member of the SGS-THOMSON Microelectronics Group.

INMOS document number: 72 TRN 219 01

Printed at Redwood Press Limited, Melksham, Wiltshire

Contents overview

1	System Products overview	1
---	--------------------------------	---

Development Systems

Software Development Tools

2	OCCam 2 Toolset	9
3	ANSI C Toolset	23
4	Glockenspiel C + +	35
5	ANSI FORTRAN 77 Toolset	43
6	ALSYS Ada Compiler	53

Transputer Development Kits

7	Transputer Development Kits	63
---	-----------------------------------	----

Systems Software

Board Support Software

8	IMS F000B VecTRAM library	69
9	IMS F001B GPIB libraries	83
10	IMS F002B SCSI libraries	93
11	IMS F003A 2D Graphics libraries	109
12	IMS F007A DSP libraries and development tools	119
13	Device Drivers and Motherboard Support Software	127
14	Network Support Software	131

Real Time Kernels and Operating Systems

15	VRTX32/T Real-time Executive	139
16	C Executive	141

Hardware Products

TRANsputer Modules (TRAMS)

17	IMS B401 32Kbyte TRAM	149
18	IMS B416 64Kbyte TRAM	155
19	IMS B410 160Kbyte TRAM	161
20	IMS B411 1Mbyte TRAM	167
21	IMS B404 2Mbyte TRAM	173
22	IMS B428 2Mbyte TRAM	181
23	IMS B417 4Mbyte TRAM	189
24	IMS B426 4Mbyte TRAM	197

25	IMS B427 8Mbyte TRAM	203
26	IMS B408 Frame store TRAM	209
27	IMS B409 Display TRAM	219
28	IMS B415 Differential link buffer TRAM	229
29	IMS B418 Flash ROM TRAM	237
30	IMS B419 Graphics TRAM	243
31	IMS B420 Vector processing TRAM	255
32	IMS B421 IEEE-488 GPIB TRAM	265
33	IMS B422 SCSI TRAM	283
34	IMS B429 Video Image Processing TRAM (VIP)	295
35	IMS B430 Prototyping TRAM	301
36	IMS B431 Ethernet TRAM	305

Motherboards and other Standard Interface Boards

37	IMS B008 IBM PC Motherboard	309
38	IMS B017 IBM PS/2 Motherboard	321
39	IMS B014 VMEbus slave card	329
40	IMS B016 VMEbus Master/Slave	333
41	IMS B015 NEC 9800 series PC Board	353
42	IMS B012 Double extended eurocard	361
43	IMS B018 TRAM motherboard	375
44	IMS B300 Ethernet connection system	377

Associated Hardware Products

45	IMS B250 VME Rack	381
46	IMS CA12 Card Frame Adapter	383
47	Cables for Board Products	385

Application Notes

48	Dual-In-Line Transputer Modules (TRAMs)	389
	(INMOS Technical Note 29)	
49	Module Motherboard Architecture	409
	(INMOS Technical Note 49)	
50	Developing parallel C programs for transputers	435
	(INMOS Technical Note 68)	

Appendices

A	Quality and Reliability	463
B	Software Licensing	465
C	Product Reference Tables	467

Indexes

	Index by product number	471
	Index by product name	472

Contents

Preface	ii
1 System Products overview	1
1.1 Introduction	2
1.2 Innovation and Quality	2
1.3 TRAMS (TRANputer Modules)	3
1.3.1 Standard Interface	3
1.3.2 Upgradability	3
1.3.3 Flexibility	4
1.3.4 Evaluation	4
1.4 Quality and Reliability	4
Development Systems	5
Software Development Tools	7
2 occam 2 Toolset	9
2.1 Introduction	10
2.2 Mapping occam Programs Onto Transputer Networks	10
2.3 Product Overview	14
2.3.1 OCCAM 2 development system	14
2.3.2 Libraries	15
occam compiler library	15
snglmath.lib, dblmath.lib	15
tbmaths.lib	15
string.lib	15
hostio.lib	15
streamio.lib	15
msdos.lib	15
crc.lib	15
convert.lib	15
xlink.lib	15
debug.lib	15
2.3.3 Mixed language programs	16
2.3.4 Debugging	16
Interactive symbolic debugging	16
Post-mortem symbolic debugging	16
T425 simulation	16
2.3.5 Optimised code generation	17
2.3.6 Assembler inserts	17
2.3.7 D700 transputer development system support	18
2.3.8 Improvements over previous releases	18
2.4 OCCAM Toolset Product Components	18
2.4.1 Documentation	18
2.4.2 Software Tools	19
2.4.3 Software libraries	19
2.4.4 Source code	19

2.5	Product Variants	19
2.5.1	IMS D7205 IBM PC and NEC PC version	19
	Operating requirements	20
	Distribution media	20
2.5.2	IMS D6205 VAX VMS version	20
	Operating requirements	20
	Distribution media	20
2.5.3	IMS D5205 Sun 3 version, IMS D4205 Sun 4 version	20
	Operating requirements	21
	Distribution media	21
2.6	Licensing Information	21
2.7	Problem Reporting And Field Support	21
3	ANSI C Toolset	23
3.1	Introduction	24
3.2	Product Overview	24
3.2.1	How programs are built	24
3.2.2	ANSI C compilation system	24
	Compiler operation	24
	ANSI conformance	24
	Optimised code generation	25
	Libraries	25
	Mixed language programs	26
	Assembler inserts	26
3.2.3	Target systems	27
3.2.4	Support for parallelism	28
3.2.5	Debugging	30
	T425 simulation	30
	Interactive symbolic debugging	30
	Post-mortem symbolic debugging	32
3.2.6	Improvements over previous releases	32
3.3	ANSI C Toolset Product Components	32
3.3.1	Documentation	32
3.3.2	Software Tools	33
3.3.3	Software libraries	33
3.4	Product Variants	33
3.4.1	IMS D7214 IBM PC version	33
	Operating requirements	33
	Distribution media	33
3.4.2	IMS D6214 VAX VMS version	33
	Operating requirements	34
	Distribution media	34
3.4.3	IMS D5214 Sun 3 version, IMS D4214 Sun 4 version	34
	Operating requirements	34
	Distribution media	34
3.5	Error Reporting And Field Support	34
4	Glockenspiel C + +	35
4.1	Product Overview	36
4.1.1	C + +	36

4.1.2	Use with the INMOS ANSI C toolset	36
4.2	Glockenspiel C++ Product Components	37
4.2.1	Documentation	37
4.2.2	Software tools	37
4.2.3	Software Libraries	37
4.3	Product Variants	37
4.3.1	IMS D7217 IBM PC version	37
	Cross-development operating requirements	37
	Native development operating requirements	38
	Distribution media	38
4.3.2	IMS D6214 VAX VMS version	38
	Cross-development operating requirements	38
	Native development operating requirements	38
	Distribution media	38
	Licensing variants	38
4.3.3	IMS D5217 Sun 3 version	39
	Cross-development operating requirements	39
	Native development operating requirements	39
	Distribution media	40
	Licensing variants	40
4.3.4	IMS D4217 Sun 4 version	40
	Cross-development operating requirements	40
	Native development operating requirements	40
	Licensing variants	40
	Distribution media	41
4.4	Error Reporting And Field Support	41
5	ANSI FORTRAN 77 Toolset	43
5.1	Introduction	44
5.2	Product Overview	44
5.2.1	How programs are built	44
5.2.2	ANSI FORTRAN compilation system	44
	Compiler operation	44
	ANSI conformance	44
	Optimised code generation	44
	Run-Time System	45
	Mixed language programs	45
5.2.3	Target systems	46
5.2.4	Support for parallelism	46
5.2.5	Debugging	48
	T425 simulation	48
	Interactive symbolic debugging	48
	Post-mortem symbolic debugging	49
5.2.6	Improvements over previous releases	50
5.3	ANSI FORTRAN Toolset Product Components	50
5.3.1	Documentation	50
5.3.2	Software Tools	50
5.4	Product Variants	50
5.4.1	IMS D7216 IBM PC version	50
	Operating requirements	51
	Distribution media	51

5.4.2	IMS D6216 VAX VMS version	51
	Operating requirements	51
	Distribution media	51
5.4.3	IMS D5216 Sun 3 version, IMS D4214 Sun 4 version	52
	Operating requirements	52
	Distribution media	52
5.5	Error Reporting And Field Support	52
6	ALSYS Ada Compiler	53
6.1	Product Overview	54
6.2	Product Highlights	54
6.2.1	Supports easy implementation of distributed Ada applications	54
6.2.2	Efficient sharing of Ada Libraries	54
6.2.3	Generates high performance, compact application code	54
	Ada Code Generation	54
	Floating point support	54
	Ada Run Time	54
	Supports low level programming features	54
6.2.4	Increased development productivity	55
6.2.5	Advanced debugging support	55
6.2.6	Ada predefined Input and Output	55
6.2.7	Transfers the loadable Ada program to the target	55
6.3	The Alsys Ada Compilation System	55
6.4	Ada Compiler Toolset Product Components	55
6.4.1	Documentation	55
	User's Guide	55
	Project Development Guide	55
	Installation Guide	55
	Ada Reference Manual	55
	Appendix F	56
	Application Developer's Guide	56
6.4.2	Software Components	57
6.5	Product Variants	57
6.5.1	IBM PC Alsys Ada Compiler	57
	Operating requirements	57
	Distribution media	57
6.5.2	VAX VMS Alsys Ada Compiler	57
	Operating requirements	57
	Distribution media	57
6.6	Customer Support And Upgrade Services	58
6.7	Alsys And Ada	58
	Alsys Offices and Addresses	59
	Transputer Development Kits	61
7	Transputer Development Kits	63
7.1	Transputer Development Kits	63

Systems Software	65
Board Support Software	67
8 IMS F000B VecTRAM library	69
8.1 Introduction	70
8.2 Product Overview	70
8.3 Using IMS F000B	70
8.3.1 Incorporation into a C program	72
8.3.2 Incorporation into an OCCAM program	72
8.4 Supplied Routines	73
8.4.1 Vector Absolute Value – Real	74
8.4.2 Vector Addition – Real	74
8.4.3 Vector Compare – Real	74
8.4.4 Disable Co-processor Error Flags	74
8.4.5 Vector Division – Real	75
8.4.6 Vector Dot Product – Complex	75
8.4.7 Vector Dot Product – Real	75
8.4.8 Enable Co-processor Error Interrupts	75
8.4.9 Fast Fourier Transform – Complex	76
8.4.10 Finite Impulse Response (FIR) Filter	76
8.4.11 Vector Floating Point to Integer (16-bit) Conversion	76
8.4.12 Infinite Impulse Response (IIR) Filter – Complex	77
8.4.13 Infinite Impulse Response (IIR) Filter – Real	77
8.4.14 Matrix Multiplication – Complex	77
8.4.15 Matrix Multiplication – Real	78
8.4.16 Inverse Fast Fourier Transform – Complex	78
8.4.17 Vector Integer(16-bit) to 32-bit floating-point Conversion	78
8.4.18 Vector Log to the base 10 – Real	79
8.4.19 Vector Magnitude – Complex	79
8.4.20 Vector Magnitude Square – Complex	79
8.4.21 Find the Element with the Maximum Value and Its Position – Real	79
8.4.22 Vector Mean – Real	79
8.4.23 Find the Element with the Minimum Value and Its Position – Real	79
8.4.24 Vector Move – Bytes	80
8.4.25 Vector Move – Words (32-bit)	80
8.4.26 Vector Multiply – Complex	80
8.4.27 Vector Multiply – Real	80
8.4.28 Vector Power – Real	80
8.4.29 Modify Co-processor Rounding Mode	81
8.4.30 Vector Scale – Real	81
8.4.31 Vector Square Root – Real	81
8.4.32 Vector Subtract – Real	81
8.5 Environment	82
8.6 IMS F000B Product Components	82
8.6.1 Distribution media	82
8.6.2 Documentation	82
8.7 Error Reporting And Field Support	82
8.8 References	82

9	IMS F001B GPIB libraries	83
9.1	Introduction	84
9.2	Product Overview	84
9.2.1	IMS F001B command format	85
	Summary of the IMS F001B commands	86
9.2.2	Using IMS F001B	87
9.3	Operating environment	92
9.4	IMS F001B Product Components	92
9.4.1	Distribution media	92
9.4.2	Documentation	92
9.5	Error Reporting And Field Support	92
10	IMS F002B SCSI libraries	93
10.1	Introduction	94
10.1.1	SCSI overview	94
10.2	Product Overview	95
10.2.1	IMS B422 Device Driver	95
10.2.2	Initialisation Interface	95
10.2.3	Initiator Mode Interface	96
	Common Command Set	96
10.2.4	Target Mode Interface	97
10.2.5	Diagnostic tests	98
10.2.6	Initialisation Interfaces	98
10.3	Incorporation into a user program	98
	Incorporation of initiator mode into a user program	98
10.4	Simple Initiator mode example	99
	Configuration	99
10.5	List of supplied procedures	103
10.6	Performance	104
10.7	Compatibility	105
10.8	Operating environment	106
10.9	IMS F002B Product Components	106
10.9.1	Distribution media	106
10.9.2	Documentation	106
10.10	Error Reporting And Field Support	107
10.11	References	107
11	IMS F003A 2D Graphics libraries	109
11.1	Introduction	110
11.2	Product Overview	110
11.2.1	Summary of the IMS F003 commands	111
11.3	Using IMS F003	115
11.4	IMS F003 Product Components	118
11.4.1	Distribution media	118
11.4.2	Documentation	118
11.4.3	Operating environment	118
11.5	Error Reporting And Field Support	118

12 IMS F007A DSP libraries and development tools	119
12.1 Introduction	120
12.2 Product Overview	120
12.2.1 Library usage	120
12.2.2 intgen – VecTRAM Interface Generation Utility	121
Running the intgen utility	121
12.2.3 ivtlink – VecTRAM Linker	122
12.3 Supplied Routines	123
12.4 Environment	126
12.5 IMS F007A Product Components	126
12.5.1 Distribution media	126
12.5.2 Documentation	126
12.6 Error Reporting And Field Support	126
12.7 References	126
12.8 Ordering Information	126
13 Device Drivers and Motherboard Support Software	127
13.1 Product overview	128
13.1.1 Device driver	128
13.1.2 INMOS server	128
13.1.3 Transputer mapping software	128
13.1.4 Switch setting support	129
13.2 Product components summary	129
13.2.1 Documentation	129
User manual	129
13.2.2 Software	129
Device driver	129
INMOS iserver (with sources)	129
Transputer mapping tool	129
Motherboard switch–setting software	129
13.3 Product variants	129
13.3.1 General operating requirements	129
13.3.2 Distribution media	129
IMS S217	129
IMS S308	129
IMS S708	129
IMS S514	129
13.4 OEM support	130
14 Network Support Software	131
14.1 Introduction	132
14.2 Product Overview	132
14.2.1 Connecting transputers to computer networks	132
14.2.2 Capabilities	132
14.2.3 System configuration	133
14.2.4 Support for INMOS development tools	133
14.2.5 Support for INMOS Ethernet connection system (IMS B300)	133
14.2.6 Support for mixed networks of machines	133
14.3 Product Component Summary	134

14.3.1	Documentation	134
	User Manual	134
14.3.2	Software	134
	Extended INMOS iserver	134
	Connection server	134
	INMOS Ethernet box configuration program	134
14.4	Product Variants	134
14.4.1	IMS S707 IBM PC	134
	Operating requirements	134
	Boards supported	135
	Distribution media	135
14.4.2	IMS S607 VAX VMS	135
	Operating requirements	135
	Distribution media	135
14.4.3	IMS S507 Sun 3 and Sun 4	135
	Operating requirements	135
	Boards supported	135
	Distribution media	135
14.5	Error Reporting And Field Support	135

Real Time Kernels and Operating Systems 137

15 VRTX32/T Real-time Executive 139

15.1	Overview	140
	Deterministic Performance	140
	Distributed Environment	140
	Support	140

16 C Executive 141

16.1	C Executive For The Transputer	142
16.2	Benefits Of C Executive	142
16.3	CE-VIEW	143
16.4	Documentation	143
16.5	Availability	143
16.6	Licensing	143
16.7	How To Order	144

Hardware Products 145

TRAnsputer Modules (TRAMS) 147

17 IMS B401 32Kbyte TRAM 149

17.1	IMS B401 TRAM engineering data	150
17.1.1	Introduction	150
17.1.2	Pin descriptions	150

17.1.3	Standard TRAM signals	151
	notError (pin 11)	151
	LinkspeedA and LinkspeedB (pins 6 and 7)	151
	Link signals	151
17.1.4	Memory configuration	151
	Location of external memory	151
17.1.5	Mechanical details	152
17.1.6	Installation	153
17.1.7	Specification	154
17.1.8	Ordering Information	154
18	IMS B416 64Kbyte TRAM	155
18.1	IMS B416 TRAM engineering data	156
18.1.1	Introduction	156
18.1.2	Pin descriptions	156
18.1.3	Standard TRAM signals	156
	notError (pin 11)	156
	LinkspeedA and LinkspeedB (pins 6 and 7)	156
	Link signals	157
18.1.4	Memory configuration	157
18.1.5	Mechanical details	158
18.1.6	Installation	158
18.1.7	Specification	159
18.1.8	Ordering Information	159
19	IMS B410 160Kbyte TRAM	161
19.1	IMS B410 TRAM engineering data	162
19.1.1	Description	162
19.1.2	Pin descriptions	162
19.1.3	Standard TRAM signals	163
	notError (pin 11)	163
	LinkspeedA and LinkspeedB (pins 6 and 7)	163
	Link signals	163
19.1.4	Memory configuration	163
19.1.5	Mechanical details	164
19.1.6	Installation	165
19.1.7	Specification	166
19.1.8	Ordering Information	166
20	IMS B411 1Mbyte TRAM	167
20.1	IMS B411 TRAM engineering data	168
20.1.1	Description	168
20.1.2	Pin descriptions	168
20.1.3	Standard TRAM signals	168
	notError (pin 11)	168
	LinkSpeedA and LinkSpeedB (pins 6 and 7)	169
	Link signals	169
20.1.4	Memory configuration	169
20.1.5	Mechanical details	170
20.1.6	Installation	170
20.1.7	Specification	171

20.1.8	Ordering Information	171
21	IMS B404 2Mbyte TRAM	173
21.1	IMS B404 TRAM engineering data	174
21.1.1	Introduction	174
21.1.2	Pin descriptions	174
21.1.3	Standard TRAM signals	175
	notError (pin 11)	175
	LinkSpeedA and LinkSpeedB (pins 6 and 7)	175
	Link signals	175
21.1.4	Subsystem signals	175
21.1.5	Memory configuration	176
	Location of external memory	176
	Subsystem register locations	176
21.1.6	Mechanical details	177
21.1.7	Installation	178
21.1.8	Specification	179
21.1.9	Ordering Information	179
22	IMS B428 2Mbyte TRAM	181
22.1	IMS B428 TRAM engineering data	182
22.1.1	Introduction	182
22.1.2	Pin descriptions	182
22.1.3	Standard TRAM signals	183
	notError (pin 11)	183
	LinkSpeedA and LinkSpeedB (pins 6 and 7)	183
	Link signals	183
22.1.4	Subsystem signals	183
22.1.5	Memory configuration	183
	Subsystem register locations	184
22.1.6	Mechanical details	185
22.1.7	Installation	186
22.1.8	Specification	187
	Notes	187
22.2	Ordering Information	187
23	IMS B417 4Mbyte TRAM	189
23.1	IMS B417 TRAM engineering data	190
23.1.1	Introduction	190
23.1.2	Pin descriptions	190
23.1.3	Standard TRAM signals	191
	notError (pin 11)	191
	LinkSpeedA and LinkSpeedB (pins 6 and 7)	191
	Link signals	191
23.1.4	Subsystem signals	191
23.1.5	Memory configuration	192
	Location of external memory	192
	Subsystem register locations	192
23.1.6	Mechanical details	193
23.1.7	Installation	194
23.1.8	Specification	195

23.1.9	Ordering Information	195
24	IMS B426 4Mbyte TRAM	197
24.1	Description	198
24.2	Pin descriptions	198
	Notes:	198
24.3	Standard TRAM signals	198
24.3.1	notError (pin 11)	198
24.3.2	LinkSpeedA and LinkSpeedB (pins 6 and 7)	199
24.3.3	Link signals	199
24.4	Memory configuration	199
24.5	Mechanical details	200
24.6	Installation	200
24.7	Specification	201
	Notes	201
24.8	Ordering Information	202
25	IMS B427 8Mbyte TRAM	203
25.1	Description	204
25.1.1	Pin descriptions	204
	Notes:	204
25.1.2	Standard TRAM signals	204
25.2	notError (pin 11)	204
25.3	LinkSpeedA and LinkSpeedB (pins 6 and 7)	205
25.4	Link signals	205
25.5	Subsystem signals	205
25.5.1	Memory configuration	205
	Subsystem register locations	205
25.5.2	Mechanical details	206
25.5.3	Installation	207
25.5.4	Specification	208
	Notes	208
25.5.5	Ordering Information	208
26	IMS B408 Frame store TRAM	209
26.1	IMS B408 TRAM engineering data	210
26.1.1	Introduction	210
26.1.2	Pin descriptions	210
26.1.3	Pixel Port signals	212
	Electrical Specification	213
26.1.4	Memory Map	213
26.1.5	Pixel Port control registers	214
26.1.6	Mechanical details	214
26.1.7	Installation	215
26.1.8	Specification	217
26.1.9	Ordering Information	217
27	IMS B409 Display TRAM	219
27.1	IMS B409 TRAM engineering data	220

27.1.1	Introduction	220
27.1.2	Pin descriptions	220
27.1.3	Pixel Bus connectors	222
27.1.4	The Pixel channels	223
	8 bits/pixel mode	223
	18 bits/pixel mode	223
	The colour look-up tables	223
	Video Outputs	223
27.1.5	Memory Map	224
	Pixel Channel Mode select	224
	The video timing generator	224
	The Colour look-up tables	224
27.1.6	Mechanical details	225
27.1.7	Installation	226
27.1.8	Specification	228
27.1.9	Ordering Information	228
28	IMS B415 Differential link buffer TRAM	229
28.1	Description	230
28.2	Pin descriptions	230
28.3	Introduction to the IMS B415	230
28.4	Principles of Operation	231
28.5	Differential Connectors	231
28.6	Cables	232
28.7	Mechanical details	233
28.8	Installation	233
28.9	Specification	234
28.10	References	236
28.11	Ordering Information	236
29	IMS B418 Flash ROM TRAM	237
29.1	Description	238
29.1.1	Bootting transputer networks with the IMS B418	238
29.1.2	Flash ROMs	238
29.1.3	The IMS B418 in the development environment	239
	Bootstrap Mode	239
	Transparent mode	239
	Auto-program mode	240
29.1.4	Firmware	240
	Summary of Programming Protocol	240
29.1.5	Programming Voltage Generator	240
29.1.6	Power-on reset / Power-fail monitor	241
29.1.7	Sub-system Pins	241
29.1.8	Link select jumpers	241
29.1.9	Mode select jumpers	241
29.2	Specifications	242
	Notes	242
29.3	Reference	242
29.4	Ordering Information	242

30 IMS B419 Graphics TRAM	243
30.1 Description	244
30.1.1 Introduction	244
30.1.2 Screen sizes	244
30.1.3 SubSystem signals	245
30.1.4 CVC reset register	245
30.1.5 Clock Select Register	245
30.1.6 Memory Map	245
30.1.7 Pixel clock selection	246
30.1.8 Jumper selection	247
30.1.9 Video and sync outputs	247
30.2 Graphics library software	248
Summary of CGI Graphics Library functions	248
30.3 Mechanical details	249
30.4 Pin descriptions	251
30.5 Specification	252
30.6 References	253
30.7 Ordering Information	253
31 IMS B420 Vector processing TRAM	255
31.1 Introduction	256
The IMS T800 Processor	256
The ZR34325 Vector/Signal Processor	256
31.2 Transputer memory map	258
31.3 ZR34325 memory map	259
31.4 Mechanical details	260
31.5 Specification	261
31.6 IMS F000 software library	262
31.6.1 Software support	262
IMS F000A function calls include:	263
31.7 Ordering Information	264
32 IMS B421 IEEE-488 GPIB TRAM	265
32.1 Description	266
32.1.1 The IEEE-488 standard	266
32.1.2 Companion software package IMS F001	266
32.2 Pin descriptions	267
32.2.1 Standard TRAM signals	267
32.2.2 Subsystem signals	268
32.3 Hardware features	269
32.3.1 Onboard transputer system	269
32.3.2 IEEE-488 Interface	269
32.3.3 Electrically Erasable Read Only Memory (EEROM)	269
32.3.4 Power-up/Power-fail detection	269
32.3.5 Jumpers	269
32.4 Connector pin assignments	270
32.4.1 IEEE-488 connector J1	270
32.4.2 Auxiliary connector J2	272

32.5	Option selection jumpers	272
32.5.1	Bus address jumpers, JP1 to JP5	272
32.5.2	Device capability jumpers, JP6 and 7	273
32.5.3	Bus drive selection jumper, JP8	273
32.5.4	Data protect jumper, JP9	273
32.6	Hardware information for programmers	274
32.6.1	Memory configuration	274
	Subsystem register locations	274
32.6.2	Input port assignments	275
32.6.3	Output port assignments	276
32.6.4	GPiB control register addresses	276
32.6.5	Wait states	277
32.6.6	Internal and external reset pulse generation	277
32.6.7	EEROM programming	277
32.7	Mechanical details	278
32.8	Installation	280
32.9	Specification	281
32.10	Ordering information	281
33	IMS B422 SCSI TRAM	283
33.1	IMS B422 SCSI TRAM engineering data	284
33.1.1	Transputer Modules (TRAMs)	284
33.1.2	Pin descriptions	284
	ClockIn	284
	LinkOut0-3	284
	LinkIn0-3	285
	LinkSpeedA, LinkSpeedB	285
	Reset	285
	Analyse	285
	notError	285
33.1.3	A Brief Description of the Small Computer System Interface	286
33.1.4	SCSI Capabilities	286
33.1.5	Connecting the IMS B422 to a SCSI bus	286
33.1.6	IMS B422 hardware	287
33.1.7	Memory map	287
33.1.8	IMS F002 board support software	288
33.1.9	Structure	288
33.1.10	IMS B422 Device Driver	289
33.1.11	Initialisation Interface	289
33.1.12	Initiator Mode Interface	289
33.1.13	Target Mode Interface	291
33.1.14	Software Distribution	291
33.1.15	Mechanical details	291
33.1.16	Installation	292
33.1.17	Specification	294
33.1.18	References	294
33.1.19	Ordering Information	294
34	IMS B429 Video Image Processing TRAM (VIP)	295
34.1	Introduction	296
34.2	Architecture	296

34.3	Supported operations	297
34.3.1	Decimation and interpolation	297
34.4	Video input circuit	297
34.4.1	Input look-up table (LUT)	298
34.5	Memory map	298
34.6	Specification	299
34.7	Ordering information	299
35	IMS B430 Prototyping TRAM	301
35.1	IMS B430 Prototyping TRAM product overview	302
	Printed Circuit Board	302
	Onboard transputer system	302
	Prototyping area	302
35.1.1	Specification	302
35.1.2	Ordering Information	303
36	IMS B431 Ethernet TRAM	305
36.1	Specification	306
36.2	Ordering Information	306
	Motherboards and other Standard Interface Boards	307
37	IMS B008 IBM PC Motherboard	309
37.1	Description	310
37.1.1	Introduction	310
37.1.2	TRAM Slots	310
37.1.3	System Services	311
37.1.4	Link Configuration	311
37.1.5	IBM PC Bus Interface	311
	Link interface	312
	Host system services	313
	DMA	313
	Interrupts	313
37.1.6	Link Speeds	313
37.2	Specifications	313
	Mechanical details	313
	Thermal Information	314
	Operating and Storage Environments	315
	Electrical details	315
37.3	Connector Pin Assignments	315
	P1 pin assignments	315
	P2 pin assignments	316
37.4	Jumpers	317
37.5	Switches	318
37.6	Reference	319
37.7	Ordering Information	319
38	IMS B017 IBM PS/2 Motherboard	321
38.1	IMS B017 engineering data	322

38.1.1	Description	322
38.1.2	TRAM Slots	322
38.1.3	System Services	323
38.1.4	Micro Channel bus interface	323
	Link interface	324
	Host system services	324
	Interrupts	324
38.1.5	Configuration	325
38.1.6	Specifications	325
	Mechanical Details	325
	Thermal Information	325
	Operating and Storage Environments	325
	Electrical Details	325
38.1.7	Memory Map	325
38.1.8	Connector Pin Assignments	326
	J3 pin assignments	326
38.1.9	References	326
38.1.10	Ordering Information	327
39	IMS B014 VMEbus slave card	329
39.1	Description	330
39.1.1	VMEbus Interface	330
39.1.2	Interrupts	330
39.1.3	IMS C004 Control	330
39.1.4	System Services Organisation	330
39.2	Specification	331
	Mechanical details	331
	Thermal details	331
	Operating and Storage Environments	331
	Electrical details	331
	VMEbus capability	332
39.3	Ordering Information	332
40	IMS B016 VMEbus Master/Slave	333
40.1	Introduction	334
	VME Bus	334
	IMS T801 Transputer	335
40.2	Description	335
40.2.1	IMS T801 and private SRAM	335
40.2.2	Primary Control Registers	335
40.2.3	MAP-RAM	336
40.2.4	Dual-Access DRAM	337
40.2.5	Byte Multiplexor	337
40.2.6	VME Features	338
40.2.7	F-ROM	339
40.2.8	Serial Ports	339
40.2.9	PEX Boards	340
40.2.10	Real-Time Clock	340
40.2.11	Resets and Transputer System Services	341
40.2.12	The Front Panel	342
40.3	'Hard' Configuration Information	343

40.4	Memory Maps	344
40.5	Specification	347
40.5.1	Mechanical and Thermal Details	347
40.5.2	Electrical Details	348
40.5.3	VMEbus capability	348
40.6	Connector Pin Assignments	349
40.7	References	351
40.8	Ordering Information	351
41	IMS B015 NEC 9800 series PC Board	353
41.1	Description	354
41.1.1	Link connections	354
41.1.2	Link speed selection	354
41.1.3	System Services	355
41.1.4	Up, Down, and Subsystem	355
41.1.5	PC interface	356
41.1.6	IO Address	356
41.1.7	Reset, Analyse and Error registers	356
41.1.8	Interface link	357
41.1.8	Interrupts	357
41.1.9	External power supplies	358
41.2	External Connections	359
41.3	Specification	360
41.4	Ordering Information	360
42	IMS B012 Double extended eurocard	361
42.1	IMS B012 Double Eurocard Motherboard engineering data	362
42.1.1	Introduction	362
42.1.2	Hardware Description	362
Link Connections	362	
P1 Links	365	
Switch Configuration Transputer	367	
Reset, Analyse and Error	368	
Link Termination	369	
Error Lights	370	
User Power Connector	371	
Uncommitted Pins	371	
42.2	Specifications	373
Mechanical Details	373	
Thermal Information	373	
42.3	Ordering Information	373
43	IMS B018 TRAM motherboard	375
Ordering Information	376	
44	IMS B300 Ethernet connection system	377
44.1	IMS B300 Network connection system engineering data	377
44.1.1	Interfaces	377

44.1.2	Diagnostic Interfaces	377
44.1.3	Protocols	378
44.1.4	Performance	378
44.1.5	Specification	378
44.1.6	Ordering information	378
Associated Hardware Products		379
45 IMS B250 VME Rack		381
45.1	Front panel	382
45.2	Specification	382
45.3	Ordering Information	382
46 IMS CA12 Card Frame Adapter		383
	Ordering Information	383
47 Cables for Board Products		385
Application Notes		387
48 Dual-In-Line Transputer Modules (TRAMs) (INMOS Technical Note 29)		389
48.1	Background	391
48.2	Introduction	392
48.3	Functional description	393
48.3.1	Pinout of size1 module	393
48.3.2	Pinout of larger sized modules	393
48.3.3	TRAMs with more than one transputer	395
48.3.4	Extra pins	395
48.3.5	Subsystem signals driven from a TRAM	395
	Subsystem registers	396
	Multiple subsystems	396
48.3.6	Memory parity	397
48.3.7	Memory map	397
48.4	Electrical description	398
48.4.1	Link outputs	398
48.4.2	Link inputs	398
48.4.3	notError output	398
48.4.4	Reset and analyse inputs	398
48.4.5	Clock input	399
48.4.6	notError input to subsystem	399
48.4.7	GND, VCC	399
48.5	Mechanical description	399
48.5.1	Width and length	399
48.5.2	Vertical dimensions	400
48.5.3	Direction of cooling	402
48.6	TRAM pins and sockets	402
48.6.1	Stackable socket pin	402

48.6.2	Through-board sockets	402
48.6.3	Subsystem pins and sockets	403
48.6.4	Motherboard sockets	403
48.7	Mechanical retention of TRAMs	403
48.8	Profile drawings	404
49	Module Motherboard Architecture	409
	(INMOS Technical Note 49)	
49.1	Introduction	411
49.2	Module motherboard architecture	411
49.2.1	Design goals	411
49.2.2	Architecture	411
49.3	Link configuration	412
49.3.1	Pipeline	412
49.3.2	IMS C004 link configuration	413
49.3.3	T212 pipeline and C004 control	413
49.3.4	Software link configuration	414
49.4	System control	415
49.4.1	Reset, analyse and error	415
49.4.2	Up, down and subsystem	415
49.4.3	Source of control	417
49.4.4	Clock	422
49.5	Interface to a separate host	422
49.5.1	Link interface	422
49.5.2	System control interface	423
49.5.3	Interrupts	423
49.6	Mechanical considerations	424
49.6.1	Dimensions	424
	Width and length	424
	Vertical dimensions	426
49.6.2	Motherboard sockets	427
49.6.3	Mechanical retention of TRAMs	427
49.6.4	Module orientation	427
49.7	Edge connectors	428
50	Developing parallel C programs for transputers	435
	(INMOS Technical Note 68)	
50.1	Introduction	437
50.2	What is the C toolset?	437
50.2.1	Introduction	437
50.2.2	Software toolset summary	437
50.2.3	Software design cycle – single transputer systems	438
	Edit	439
	Compile	439
	Linking	439
	Boot Strap	439
	Run	439
	Debugging	439
50.2.4	Software design cycle – multiple transputer systems	440
	Configure	441

50.3	Example problem description	441
50.3.1	Introduction	441
50.3.2	What is configuration?	441
50.4	Using the Dx214 C toolset	442
50.4.1	Introduction	442
50.4.2	C parallel processing library extensions	442
50.4.3	Parallel version on one transputer, all in C	443
	Setting up a process	444
	Running the processes in parallel	445
	Channel communications	445
	The master process	446
	The worker process	447
	Building the upper case example	448
	Running the single processor version	448
50.4.4	Configuring a multi-processor version using icconf	449
	Introduction	449
	Introducing some new tools	449
	The master process	450
	The worker process	451
	Connecting to the external configuration channels	451
	Writing a configuration script	452
	Declaring the physical hardware	454
	Showing physical interconnect	454
	Interface description	454
	Wiring things up	455
	Pulling in the real code	455
	Placing the processes onto processors	455
	Building the example	455
	Running the multi-processor version	456
50.5	Conclusions	456
50.6	Differences between 3L and icc concurrency library	457
	Upgrading code to IMS Dx214 from IMS D711	457
50.7	Useful hints when writing Dx214 C toolset programs	458
	Program design methodology	458
50.7.1	What if the program will not run?	458
50.8	References	459
Appendices		461
A Quality and Reliability		463
B Software Licensing		465
	End User Licences	465
	Distribution Licences	465
C Product Reference Tables		467
C.1	Table of Composite Products	467
C.2	Table of Board Products and Associated Software	467
Indexes		469

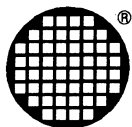
Preface

Development tools and system products are important and developing areas of application for INMOS devices. *The Transputer Development and iq Systems Databook* has been published to provide detailed information on the INMOS product range.

The databook comprises an overview, engineering data and applications information for the current range of development tools and systems products.

INMOS provide a wide range of development tools including compilers, toolsets and development kits. A diverse range of software is also available. INMOS systems products provide powerful development platforms for system designers interested in high density, high performance, design simplicity and cost effectiveness.

In addition to development tools and systems products, the INMOS product range also includes transputer products and high performance graphics devices. For further information concerning INMOS products please contact your local SGS-THOMSON sales outlet.



Systems Products overview

1.1 Introduction

The SGS-THOMSON Microelectronics Group is a major supplier of a wide range of semi-conductor devices and commands leading market positions in many product areas. The acquisition of INMOS has strengthened SGS-THOMSON's portfolio. INMOS, based in the UK, operates in the same way as all other divisions of SGS-THOMSON and services its customers through the SGS-THOMSON sales network.

INMOS is a recognised leader in the development and design of high-performance integrated circuits and is a pioneer in the field of parallel processing. Components are manufactured and designed to satisfy the most demanding of current applications and also provide an upgrade path for future applications. Current designs and development will meet the requirements of systems in the next decade. Computing requirements essentially comprise high performance, flexibility and simplicity of use. These characteristics are central to the design of all INMOS products.

INMOS has a consistent record of innovation over a wide product range and, together with its parent company SGS-THOMSON Microelectronics, supplies components to system manufacturing companies in the USA, Europe, Japan and the Far East. INMOS products include a range of microprocessors called transputers, a range of systems products known as *iq* systems and high performance graphics devices. The corporate headquarters, product design team and worldwide sales and marketing management are based at Bristol, UK.

The SGS-THOMSON Microelectronics Group is constantly upgrading, improving and developing its product range and is committed to maintaining a global position of innovation and leadership.

The Transputer Development and iq Systems Databook has been published to assist in the choice of transputer development support products available for SUN, VAX and PC users and to give detailed information on the range of INMOS *iq* systems products specifically designed for integration into end-user systems. INMOS *iq* systems business is dedicated to supplying and servicing the systems builder with innovative and high quality modular products.

Design engineers will find it convenient to use this book in conjunction with *The Transputer Databook*, one of a series of product specific databooks which detail the INMOS product range.

1.2 Innovation and Quality

The INMOS transputer family of microprocessors is the industry standard in the field of multi-processing. The family consists of a range of powerful VLSI devices which all adhere to the same basic architecture incorporating a processor, memory and communication links for direct connection to other transputers.

Multiprocessor systems can be constructed from a collection of transputers operating concurrently and communicating through links. Unlike all other microprocessor implementations currently commercially available, additional bus arbitration logic is not required.

INMOS provides a wide range of tools to support development on the transputer. These have been designed to enable users to easily evaluate transputers and develop systems smoothly within the shortest possible timescales. Development tools include C, FORTRAN and PASCAL compilers and development packages based on the OCCAM compiler. A wide range of software is also available from third parties, including products such as Ada, VRTX and C Executive, overviews for which are included in this databook. Details of other third party software products for the transputer may be found in *The Transputer White Pages Directory*, available from your local SGS-THOMSON sales outlet. INMOS provides technical field support, software training courses and a comprehensive software support service.

INMOS has further exploited the power of the transputer architecture and technology by providing a range of modular hardware products for integration into end-user systems and for use as development platforms for general transputer projects. These TRAMs (TRANsputer Modules) are part of the total INMOS *iq* systems strategy to fully support the systems builder in terms of innovation and quality for INMOS products and service.

Technical expertise in design and manufacturing of transputer silicon and associated software provides an excellent basis for a professional system product range. INMOS has been a successful supplier of

silicon products to the electronics industry for many years and fully recognises the importance of service and quality in the high technology business sectors. In a demanding and competitive marketplace INMOS is fully aware of the critical importance to the system builder of reliable products and effective supplier support.

The experience, expertise and innovation of INMOS combined with the full support and resources of its parent multinational company, SGS-THOMSON Microelectronics, results in a stable yet efficient business foundation.

1.3 TRAMS (TRAnsputer Modules)

The INMOS TRAM concept was introduced during 1987 to exploit some of the major benefits of the transputer and parallel processing.

TRAMS are small, cost effective sub-assemblies of transputers and other circuitry (often RAM) with a very simple but efficient 16 signal interface standard profiled in modular sizes. The interface accommodates 4 serial INMOS links for inter-processor communication, power supply and system signals.

With this standard, TRAMs may be mounted onto a variety of motherboards which provide specific host interface hardware. Each motherboard can connect to a number of TRAMs and provides facilities for configuring a network of TRAMs to the user specified topology under software control. A software package is provided for motherboards which allows this task to be undertaken with the minimum of effort.

The TRAM architecture offers many advantages over conventional system configurations. The following features are included:

- An industry standard yet simple multiprocessor interface
- Upgradability at incremental cost
- Maximum flexibility of cost/performance with minimal real-estate

This modular philosophy will continue to be enhanced with new products on current and future silicon technology.

1.3.1 Standard Interface

The electrical interface to every TRAM product consists of 16 signal pins meeting a standard electrical and mechanical format. All TRAMs are based upon a single module profile with a defined pin layout. This single format is known as Size 1. Larger TRAMs are simply multiple sizes of this format with the same pin spacing. This published format will be maintained for future INMOS TRAM development and has already been adopted by many third party developers to extend the range of TRAM options and hosts.

1.3.2 Upgradability

Upgradability has been a major attraction for customers both for development and end product applications. TRAMs are becoming known as a futureproof solution. Investment made today using TRAMs, with respect to software and hardware engineering, can be regarded as an investment for future designs. The transputer's unique ability to distribute application software by booting networks via the links enables TRAMs to fully exploit current hardware technology and, at the same time, liberate software development from hardware constraints of cost, real-time performance and compute power. For example, a system designed today by the system integrator may be very efficiently enhanced at some time in the future as a result of a change in his end market demand. This can be achieved in two ways:

- Replace existing TRAMs with faster transputer TRAMs as they become available (eg, T805-25 to T805-30)
- Add additional TRAMs to the existing hardware

In either case the system cost is incremental and it is possible to operate with the same original application software by simply reconfiguring and booting the new network. Using traditional sequential multiprocessor solutions the second approach would inevitably result in a complete system hardware and software re-design, significant expansion in board area and a drawn out time to market.

1.3.3 Flexibility

Many system designers exploit the modularity of TRAMs to provide a range of products meeting varieties of performance/cost demand mix. For example, due to the modularity of the hardware and software, a customer may develop a low budget product and a high performance product from the same range of components. In addition, the same design can be used across the INMOS range of motherboards or specific customer designed motherboards which conform to the TRAM specification. This results in a single design being able to exploit a wide range of host environments and markets.

Unlike other architectural implementations, this flexibility can be achieved utilising just one application software package. A further advantage of this approach is the commonality of TRAM components within each end product type. This offers the system builder significant savings by minimising inventory holding.

1.3.4 Evaluation

Customers investing in the TRAM architecture for transputer evaluation purposes have the opportunity to immediately investigate the performance and characteristics of new transputer silicon as it becomes available.

1.4 Quality and Reliability

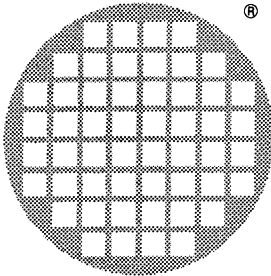
A description of the SGS-THOMSON Microelectronics approach to Quality and Reliability is included later in this databook. The subject is comprehensively detailed in the SGS-THOMSON Quality and Reliability publication *SURE 5*. This program is applied totally to all INMOS products.



Development Systems



Software Development Tools



inmos®

IMS D7205
IMS D6205
IMS D5205
IMS D4205

occam 2 Toolset

Product Information

occam cross-development systems for IBM PC, NEC PC,
Sun 3, Sun 4 and VAX hosts

KEY FEATURES

- occam compiler for INMOS transputer family of processors
- Tools to support separate compilation (linker, librarian)
- Tools for creating and loading multiprocessor programs
- Breakpointing and post-mortem symbolic debugger for multiprocessor programs
- Automatic make file generator
- T425 transputer simulator
- Scientific libraries
- Support for all types of transputer (32 and 16 bit)
- Support for mixed networks containing different transputer types
- Support for assembler inserts in OCCAM programs
- Support for mixed language programming
- Tools to support preparation of programs for EPROM
- Consistent tools across PC, VAX VMS, Sun 3, and Sun 4 hosts

APPLICATIONS

- Embedded systems (both single and multiple transputers)
- Evaluation of transputers for concurrent applications
- Scientific programming
- Education in concurrent programming

2.1 Introduction

The OCCam programming language is a high-level language which supports the design and implementation of concurrent systems on networks of processors. The OCCam language is described in the following publication:

OCCam 2 Reference Manual, published by Prentice-Hall, ISBN 0-13-629312-3.

The INMOS OCCam toolset provides a complete OCCam cross-development system for the INMOS transputer family of microprocessors. The toolset can be used to build parallel programs for single transputers and for multi-transputer networks consisting of arbitrary mixtures of transputer types. Programs developed with the toolset are both source and binary compatible across all host development machines. The INMOS OCCam toolset is available for four development platforms:

IMS D7205	OCCam toolset for IBM and NEC PC under DOS	Single user licence
IMS D6205	OCCam toolset for VAX VMS	Multi-user licence
IMS D5205	OCCam toolset for Sun 3 under SunOS	Multi-user licence
IMS D4205	OCCam toolset for Sun 4 under SunOS	Multi-user licence

This data sheet gives a technical overview of the OCCam toolset. Firstly a number of examples are shown to illustrate how parallel programs and multi-processor programs can be written using the OCCam language. An overview of the tools and libraries provided in the toolset is given, followed by a summary of the documentation and software components of the toolset. A list of the major improvements over the previous OCCam toolset product is included. Finally the operating requirements and distribution media for each of the PC, VAX and Sun products are specified, together with details on the licensing and support of the products.

2.2 Mapping OCCam Programs Onto Transputer Networks

The OCCam programming model consists of parallel processes communicating through channels. Channels connect pairs of processes and allow data to be passed between them. Each process can be built from a number of parallel processes, so that an entire software system can be described as a process hierarchy. Each channel has a `PROTOCOL` which determines the types of messages that may flow across it. This model is consistent with many modern software design methods.

Figure 2.1 shows a collection of four processes communicating over channels. The `multiplexor` process communicates with a host computer and hands out work to be done to one of three `worker` processes. Results from the workers are then returned to the host by the multiplexor. The following examples show how this collection of processes can be described in OCCam and how the OCCam program can be mapped onto a network of processors.

The OCCam example in Figure 2.2 illustrates how to program the collection of parallel processes in Figure 2.1. It assumes that the `multiplexor` and `worker` processes have been compiled and the code has been placed in the transputer code files `mux.tco` and `worker.tco`.

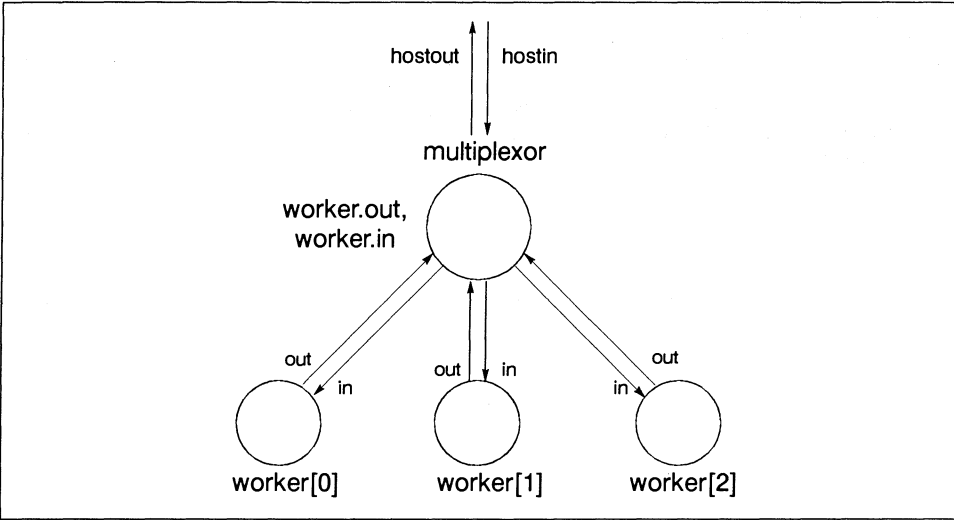


Figure 2.1 Software Network

```
#INCLUDE "hostio.inc" -- contains SP protocol definition
#USE "mux.tco"
#USE "worker.tco"
PROC example (CHAN OF SP hostinput, hostoutput, [ ]INT memory)
  [3]CHAN OF SP worker.in, worker.out:
  PAR
    multiplexor(hostinput, hostoutput, worker.out, worker.in)
  PAR i = 0 FOR 3
    worker(worker.in[i], worker.out[i])
  :
```

Figure 2.2 Parallel Processes in OCCAM

The OCCAM compiler in the toolset can be used to compile the program shown in Figure 2.2 and produce a code file to run on a single transputer. Alternatively it may be desirable to distribute the program over a network of processors. The program can be mapped onto a network using the configuration tools. A *configuration language* is used to describe the transputer network, and the mapping of the OCCAM program onto the transputer network.

The following two examples illustrate how to configure the example program for a transputer network. In both cases we assume the `multiplexor` and `worker` processes in the software network have been compiled and linked into files `mux.lku` and `worker.lku` respectively. Two hardware networks are considered: the single processor network shown in Figure 2.3 and the four-processor network shown in Figure 2.5.

Figure 2.4 shows the configuration text for mapping the software network in Figure 2.1 onto the hardware shown in Figure 2.3: a single T800 with 1 Mbyte of memory, connected to the host by link 0. In this example all the processes run on the root transputer. It might be useful to test this configuration before moving to a four-processor network.

Figure 2.6 shows the configuration text for mapping the software network on to the hardware shown in Figure 2.5; four T800s, each with 1 Mbyte of memory. In this example the `multiplexor` process runs on the root transputer while the individual `worker` processes run on each of the other transputers.

Each configuration example includes a NETWORK description section which describes how a particular network is connected. Since many applications can be run on identical networks it may be appropriate to include this definition from a standard file. The NETWORK description is followed by a CONFIG section which is virtually identical to the parallel process structure shown in Figure 2.2. The parallel process structure has been extended with PROCESSOR directives indicating which processor is to run each process. The description of the software network is almost identical in both cases; only the allocation of the worker processes is different.

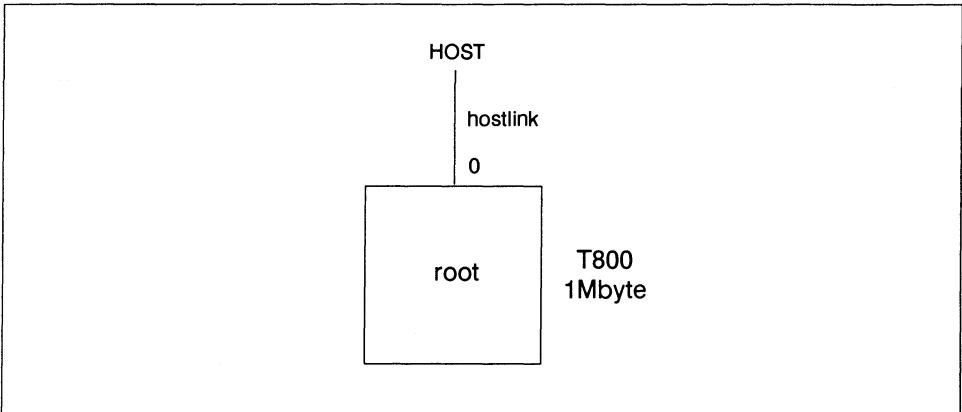


Figure 2.3 Hardware Network 1

```

VAL M IS 1024*1024:
NODE root:
ARC hostlink:
NETWORK ONE.PROCESSOR
  DO
    SET root(type, memsize := "T800", 1*M)
    CONNECT root[link][0] TO HOST WITH hostlink
  :
#INCLUDE "hostio.inc"
#USE "mux.lku"
#USE "worker.lku"
CONFIG
  [3]CHAN OF SP worker.in, worker.out:
  CHAN OF SP hostinput, hostoutput:
  PLACE hostinput, hostoutput ON hostlink:
  PAR
    PROCESSOR root
      multiplexor(hostinput, hostoutput, worker.out, worker.in)
    PAR i = 0 FOR 3
      PROCESSOR root
        worker(worker.in[i], worker.out[i])
  :

```

Figure 2.4 Software Configuration 1

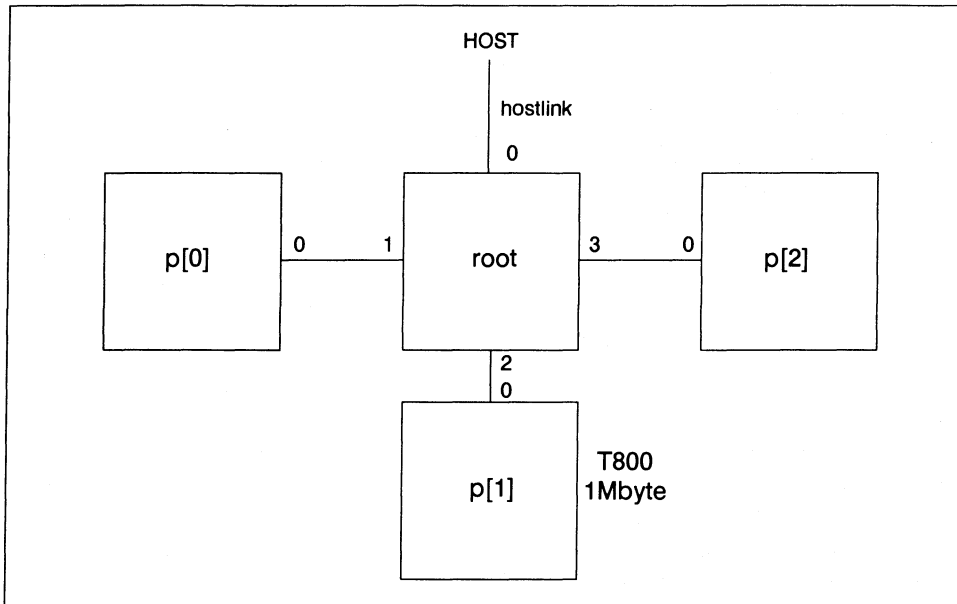


Figure 2.5 Hardware Network 2

```

VAL M IS 1024*1024:
NODE root:
[3]NODE p:
ARC hostlink:
NETWORK FOUR.PROCESSOR
DO
  SET root(type, memsize := "T800", 1*M)
  CONNECT root[link][0] TO HOST WITH hostlink
  DO i = 0 FOR 3
    DO
      SET p[i](type, memsize := "T800", 1*M)
      CONNECT root[link][i+1] TO p[i][link][0]
:
#include "hostio.inc"
#USE "mux.lku"
#USE "worker.lku"
CONFIG
  CHAN OF SP hostinput, hostoutput:
  PLACE hostinput, hostoutput ON hostlink:
  [3]CHAN OF SP worker.in, worker.out:
  PAR
    PROCESSOR root
      multiplexor(hostinput, hostoutput, worker.out, worker.in)
    PAR i = 0 FOR 3
      PROCESSOR p[i]
        worker(worker.in[i], worker.out[i])
:
    
```

Figure 2.6 Software Configuration 2

2.3 Product Overview

2.3.1 OCCAM 2 development system

The OCCAM 2 compiler supports the full OCCAM 2 language as defined in the OCCAM 2 reference manual. In addition to type checking the compiler will check programs to ensure that variables and communication channels are being used correctly in parallel components of a program. This detects many simple programming errors at compile time. The language provides support for low-level programming, including the allocation of variables to specific memory addresses, and the inclusion of transputer assembly code.

The compiler will generate code for the full range of transputer types; IMS M212, T212, T222, T225, T400, T414, T425, T800, T801, and T805. Since the different processor types share a common core of instructions it is possible to create compiled code which will run on a range of processors.

A program may be compiled in one of several 'error modes' which determine the behaviour of the program when a run-time error occurs. A mode which supports the use of the post-mortem debugger may be chosen; in this mode the network will come to a halt when a run-time error occurs. A compiler option allows all error checking to be removed from time-critical or proven parts of a program, and unchecked parts can be called from within a checked system.

The processor target, error mode and other compilation options are specified by command line switches.

Collections of procedures and functions can be compiled separately with the OCCAM 2 compiler. Separately compiled units may be collected together into libraries. The linker is used to combine separately compiled units into a program to run on a single processor. The linker supports selective loading of library units, and the linking of components written in other programming languages.

The toolset supports the use of **make**, a program building tool available under UNIX and other operating systems. The input to **make** is a 'Makefile' which describes how a program is to be built from its parts, and **make** rebuilds those parts of the program which have changed. In the OCCAM toolset a tool called **lmakef** is provided which will generate a Makefile automatically from the OCCAM program text. This guarantees that the Makefile is consistent with the program sources.

To create a multi-transputer program, the mapping of the processes to processors can be described in the configuration description. This description is processed by a pair of tools called the *configurer* and the *collector*. The configurer checks the description, and passes information to the collector on how the program code and data should be mapped onto the network. The collector creates the bootstrap and routing information necessary to load the entire network, and stores this, along with the compiled code, in a program code file. The program code file can be sent to one transputer in a network, and the program will automatically boot all the processors in the network and distribute the application code to them.

A server program on the host, called the **lserver**, can be used to load programs on to transputer networks. Once loaded, the programs start automatically. The server supports access to the host terminal and file system from the transputer network.

A compiled and configured program may be run on a network under the control of an interactive debugger. One processor in the network is used to run the debugger. Breakpoints may be set in the program on any processor in the network. Processes stopped at breakpoints may be examined and continued.

As an alternative to loading the program code from a host, the program code may be placed into an EPROM. The program code for a whole network of processors may be booted and loaded from a single processor with a ROM. The program code file may be converted to an industry standard format for programming EPROMs, using the EPROM tools in the toolset.

If the transputer network halts because of a run-time error, or if the user interrupts the server, the symbolic network debugger may be used, in post-mortem mode, to investigate the state of the network in terms of the program source text.

A transputer network may consist of any combination of processor types; a network containing a number of different types is called a *mixed network*. The configuration tools, EPROM support tools and debugging tools all support mixed networks.

2.3.2 Libraries

The OCCAM toolsets provide a wide range of OCCAM libraries, including mathematical functions, string operations and input/output functions. The libraries support similar functions across the full range of transputer types, making it easy to port software between transputer types. Sources of most of the libraries are provided, for adaptation if required.

The libraries provided are listed below.

Occam compiler library

This is the basic OCCAM run-time library. It includes: multiple length arithmetic functions; floating point functions; IEEE arithmetic functions; 2D block moves; bit manipulation; cyclic redundancy checks; code execution; arithmetic instructions. The compiler will automatically reference these functions if they are required.

snglmath.lib, dblmath.lib

Single and double length mathematics functions (including trigonometric functions). These libraries use floating point arithmetic and will produce identical results on all processors. The OCCAM source code is also provided.

tbmaths.lib

Mathematical functions optimised for the T414, T425 and T400 processors. These functions provide slightly different results to the maths library above but within the accuracy limits of the function specifications. The OCCAM source code is also provided.

string.lib

String manipulation procedures. The OCCAM source code is also provided.

hostio.lib

Procedures to support access to the host terminal and file system through the file server. The OCCAM source code is also provided.

streamio.lib

Procedures to read and write using input and output streams. The OCCAM source code is also provided.

msdos.lib

Procedures to access certain DOS specific functions through the file server. The functions are specific to the IBM PC. The OCCAM source code is also provided.

crc.lib

Procedures to calculate the cyclic redundancy check (CRC) values of strings.

convert.lib

Procedures to convert between strings and numeric values.

xlink.lib

Procedures implementing link communications which can recover after a communication failure.

debug.lib

Procedures supporting the insertion of debugging messages and assertions within a program, for use with the interactive debugger.

2.3.3 Mixed language programs

It is often appropriate in the development of a large system to use a mixture of programming languages. For example, it may be necessary to preserve an investment in existing C code, while using OCCaM to express the concurrency and communication within the system.

The OCCaM programming model provides an excellent vehicle for building mixed language systems where the components built in each language can be clearly defined with a simple interface. The OCCaM toolset can be used to bind these components together and distribute them over a network of transputers.

The OCCaM toolset supports calling C functions directly from OCCaM. It is possible to call C functions which require static data or heap; some OCCaM procedures are provided to set up OCCaM arrays for use as static or heap area for collections of C functions.

The associated INMOS ANSI C toolset allows OCCaM procedures and (single valued) OCCaM functions to be called from C just like other C functions. In addition OCCaM programs can be mixed with C programs at the configuration level.

2.3.4 Debugging

The OCCaM toolset provides three debugging tools: an interactive symbolic debugger, a post-mortem symbolic debugger, and an interactive T425 simulator.

Interactive symbolic debugging

The interactive debugger provides source level interactive debugging of a program running on a network of processors. The debugger supports breakpoints at the OCCaM or C source code level. Breakpoints may be set on any processor in the network. The state of any halted process in the network can be examined symbolically. Variables – both scalar and arrays – may be read or written symbolically. The call stack can be examined to determine the nesting of procedure calls and parallel process instantiation. Channels can be inspected. If another process is waiting on a channel it is possible to inspect the state of the waiting process, even if it is on another processor. The debugger will automatically switch between OCCaM and C when debugging a mixed language program.

The interactive debugger also provides low-level examination of memory on any processor in the network.

Post-mortem symbolic debugging

The post-mortem debugger can be used to examine the state of a transputer network symbolically. The debugger works with exactly the same code as will run in the final product; there is no additional code inserted to support debugging. This allows debugging in those circumstances where the program works under simulation or interactive debugging, but fails when run normally. After a program has halted or been interrupted by the developer, the state of the network can be preserved so that the post-mortem debugger can be run. The post-mortem debugger will support direct analysis of the network, or allow the state of the network to be saved in a dump file for later analysis. The post-mortem debugger supports the same symbolic and machine level browsing functions as the interactive debugger.

Both the interactive debugger and the post-mortem debugger run on a transputer. When doing interactive debugging, a processor must be allocated to the debugger, in addition to the target network. When doing post-mortem debugging, the state of one of the processors must be saved before running the post-mortem debugger on it.

T425 simulation

The T425 simulator is a simulation of the IMS T425 processor connected to a host running the **lserver**. It allows transputer programs to be executed on the host machine (except in the case of the IBM PC toolset which uses a transputer board to run the simulator), and supports machine level debugging of transputer code.

The machine level debugging support includes: breakpoints and single stepping at machine code level, browsing memory in different forms including disassembled machine code, access to registers and processor queues.

The transputer simulator can be used to run transputer programs on the Sun 3, Sun 4 or VAX. Code for a transputer network can usually be re-configured for a single processor (as shown in section 2), allowing the simulator to be used to try out multi-processor programs as well as single processor programs.

A batch mode is provided for running test suites.

2.3.5 Optimised code generation

The OCCAM compiler implements a wide range of code optimisation techniques:

Constant folding. The compiler evaluates all constant expressions at compile time.

Workspace allocation. Frequently used variables are placed at small offsets in workspace, thus reducing the size of the instructions needed to access them, and hence increasing the speed of execution.

Dead-code elimination. Code that cannot be reached during the execution of the program is removed.

Peephole optimisation. Code sequences are selected that are the fastest for the operation.

Instruction scheduling. Where possible the compiler exploits the internal concurrency of the transputer. In particular integer and floating point operations can be overlapped to exploit the parallel execution of the integer processor and floating point processor on the T800 series.

Constant caching. Some constants have their load time reduced by placing them in a constant table.

CASE statements. The compiler can generate a number of different code sequences to cover the dense ranges within the total range.

Inline code Procedures and functions can optionally be implemented as inline code.

The compiler and linker provide features which allow programmers to make good use of the transputer's on-chip RAM.

2.3.6 Assembler inserts

The OCCAM toolset provides an assembler insert facility. The assembler insert facility supports

- Access to full instruction set of the transputer
- Symbolic access to OCCAM variables
- Pseudo-operations to load multiple values into registers
- Ability to load results of OCCAM expressions to registers
- Labels and jumps
- Directives to access particular workspace values

2.3.7 D700 transputer development system support

The D700 transputer development system (TDS) is a fully integrated development system for OCCam and transputers. Some software is provided with the OCCam toolset to support migration from the D700:

- D700 folded file flattening tools.
- Tool for altering library usage directives to toolset style.
- Libraries to support existing D700 functions in toolset programs.

2.3.8 Improvements over previous releases

The D7205, D6205, D5205, and D4205 OCCam toolsets represent a considerable improvement over the D705B, D605A and D505A toolset releases. A summary is provided below.

- Sun 4 host is now supported.
- Compatible with the new INMOS ANSI C toolset release (IMS D7214, D6214, D5214, D4214): identical object format and many tools in common.
- Improved mixing of C and OCCam.
- Faster execution.
- Multiple error detection by the compiler rather than by a separate tool.
- Interactive debugging on the target network, including breakpoints, and the ability to examine and modify the state of processes.
- Improved configuration language, supporting separate descriptions of hardware and software, and automatic placement of channels on links.
- Inline procedures and functions.
- Improved assembler insert support.
- Full EPROM support included in toolset.
- Improvements in runtime libraries.

With the longer term in mind the object file format for this release has been changed from previous releases. Conversion tools are provided to convert old format object files into the new format, but to use the new features of the toolset existing code should be recompiled into the new format.

2.4 OCCam Toolset Product Components

2.4.1 Documentation

- Delivery manual
- User manual
- Toolset handbook
- OCCam 2 language reference manual
- Tutorial introduction to OCCam

2.4.2 Software Tools

- **oc, llink, llibr**: OCCam compiler, linker and librarian
- **lmakef, llist**: Makefile generator and binary lister program
- **occonf, icollect**: configuration tools
- **lsim, ldebug, lskip, ldump**: debugging tools
- **lserver**: INMOS host server program
- **leprom, lemit**: EPROM and memory interface programming tools
- **lcvlink, lcvemit**: conversion tools for old file formats
- **lflat, lflist, ldirect**: TDS conversion tools

2.4.3 Software libraries

- OCCam compiler libraries
- **snglmath, dblmath**: mathematics functions (includes sin, cos, etc.)
- **tbmaths**: mathematics functions optimised to run on T414, T425 and T400
- **string**: string manipulation procedures
- **hostlo**: file and terminal i/o procedures
- **streamio**: file and terminal stream i/o procedures
- **msdos**: access to certain MS-DOS calls
- **crc**: CRC calculation procedures
- **convert**: string-number conversion procedures
- **xlink**: extraordinary link handling procedures
- **debug**: debugging procedures.
- **streamco**: TDS conversion library

2.4.4 Source code

- OCCam example programs
- Source of makefile generator
- Source of server program
- OCCam library sources

2.5 Product Variants

2.5.1 IMS D7205 IBM PC and NEC PC version

Although the PC tools are invoked as if they were ordinary PC resident tools, they actually run on a transputer board plugged into the PC. A number of the tools are additionally provided in a form which will run directly on the PC.

Operating requirements

You will need one of the following configurations:

- IBM PC XT or AT with 512Kbyte memory, an IMS B008 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3)
- NEC PC-9801 with 512Kbyte memory, an IMS B015 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3).

In each case you will require:

- DOS 3.0 or later
- 9 Mbyte of free disk space

The interactive symbolic debugger requires a 2 Mbyte IMS B404 TRAM in addition to the target network.

Distribution media

Software is distributed on two media systems (both of which are supplied in the product):

- 360 Kbyte (48TPI) 5.25 inch IBM format floppy disks
- 720 Kbyte 3.5 inch IBM format floppy disks.

2.5.2 IMS D6205 VAX VMS version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Operating requirements

For hosted cross-development you will need:

- VAX VMS 5.0 or later
- 10 Mbytes of free disk space.

For loading target systems and target debugging you will need one of the following:

- A third party interface board supporting a connection to a 32 bit transputer with 2 Mbytes of memory
- An IBM PC cross development system, plus DECNET connection.

Distribution media

Software is distributed on a TK50 tape cartridge in VMS backup format.

2.5.3 IMS D5205 Sun 3 version, IMS D4205 Sun 4 version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Operating requirements

For hosted cross-development and debugging you will require:

- A Sun 3 or Sun 4 workstation or server
- SunOS 4.0.3 or later
- 10 Mbytes of free disk space.

For loading target systems and remote debugging you will require one of the following:

- IMS B014 VME motherboard with 2 Mbyte IMS B404 TRAM
- IBM PC development system plus PC-NFS.
- IMS B016 VME master board, (a very high performance interface of up to 4 transputer networks can be constructed using this board).

The interactive symbolic debugger requires a 2 Mbyte IMS B404 TRAM in addition to the target network.

Distribution media

Sun 3 software is distributed on DC600A data cartridges 60 Mbyte, QIC-11, tar format.

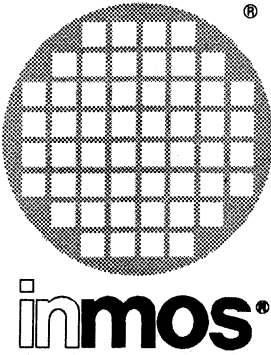
Sun 4 software is distributed on DC600A data cartridges 60 Mbyte, QIC-24, tar format.

2.6 Licensing Information

Multi-user licences are available for the D4205 (Sun 4 version), D5205 (Sun 3 version), and D6205 (VAX VMS version). No licence fee is charged for including the binary of the INMOS libraries in software products. Example programs, and other sources provided, may be included in software products. Full licensing details are available from SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.

2.7 Problem Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates. Software problem report forms are included with the software. INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.



IMS D7214
IMS D6214
IMS D5214
IMS D4214
ANSI C Toolset

Product Information

ANSI C cross-development systems for IBM PC, NEC PC, Sun 3, Sun 4 and VAX hosts

KEY FEATURES

- Full ANSI C compiler (X3.159-1989)
- Validated against Plum Hall validation suite
- Excellent compile time diagnostics
- Optimised code generation
- Support for parallelism
- Support for all INMOS processors (32 and 16 bit)
- Interactive symbolic debugging for transputer networks
- Post-mortem symbolic debugging for transputer networks
- T425 simulator
- Support for assembler inserts
- Support for EPROM programming
- Consistent tools across PC, VAX VMS, Sun 3, and Sun 4 hosts

APPLICATIONS

- Embedded systems (both single and multiple transputers)
- Porting of existing software and packages
- Evaluation of transputers for concurrent applications
- Scientific programming

3.1 Introduction

The following discussion outlines the use and features of the ANSI C toolset in moderate technical detail. In particular examples are included to illustrate how easily parallel programs and multi-processor programs can be built using the ANSI C toolset. A summary of the documentation and tools provided in the toolset is given. Finally the operating requirements and distribution media for each of the PC, VAX and Sun products are specified.

3.2 Product Overview

The INMOS C Toolsets provide complete C cross-development systems for transputer targets. They can be used to build parallel programs for single transputers and for multi-transputer networks consisting of arbitrary mixtures of transputer types. Programs developed with the toolset are both source and binary compatible across all host development machines. The INMOS ANSI C Toolset is available for four development platforms:

IMS D7214 ANSI C Toolset for IBM and NEC PC under PCDOS

IMS D6214 ANSI C Toolset for VAX VMS

IMS D5214 ANSI C Toolset for Sun 3 under SunOS 4.0.3

IMS D4214 ANSI C Toolset for Sun 4 under SunOS 4.0.3

3.2.1 How programs are built

C programs may be separately compiled. Separately compiled units may optionally be collated into libraries. The linker links separately compiled units and libraries into fully resolved main program units. Collections of communicating main program units may be distributed across networks of transputers using the configuration tools. The results of configuration may be either loaded down a transputer link using the host server or converted to an industry standard EPROM format for programming EPROMs. In addition to loading programs down a transputer link, the host server program provides access to host operating system facilities through a remote procedure call mechanism. This method is used to support the full ANSI C run time library. The server program is provided in C source code to allow extension by developers. For developers using a make program, a tool **lmakef** is provided which will generate a Makefile automatically. This automates the build process and guarantees that the Makefile is consistent with program sources.

3.2.2 ANSI C compilation system

Compiler operation

The compiler operates from a host command line interface. The preprocessor is integrated into the compiler for fast execution. The compile time diagnostics provided by the compiler are truly excellent. These include type checking in expressions and type checking of function arguments.

ANSI conformance

The INMOS ANSI C Toolset supports the full standard language as defined in X3.159-1989. The key extensions and functions which are in the ANSI standard but not in the original definition of C by Kernighan and Ritchie are:

- Prototypes to define function parameters.
- Better definition of the preprocessor.
- At least 31 characters are significant in identifiers. For external names, the limit is 6 characters. The ANSI C toolset implementation allows arbitrarily long identifiers, the first 256 characters of which are significant.

- Trigraphs introduced for use by some character sets which do not have some of the normal C characters.
- Data items can be declared **const** or **volatile**.
- Support for special character sets (especially Asian ones).
- Enumerations introduced.
- Implementation limits accessible from header files.
- Structures can be assigned, passed to functions and returned from them.

The compiler passes all the tests in the validation suite from Plum Hall. The compiler will be validated by the British Standards Institution (BSI). The validation will be recognised across Europe by the French (AF-NOR) and Italian (IMQ) Standards Institutes. The USA National Institute of Standards and Technology (NIST) is expected to recognise the validation in due course.

Optimised code generation

The ANSI C toolset achieves up to a 15 to 20 percent improvement in program execution speed over previous INMOS C compilers as measured by an internal set of program benchmarks. The compiler implements a wide range of code optimisation techniques.

Constant folding. The compiler evaluates all integer and real constant expressions at compile time.

Workspace allocation. Frequently used variables are placed at small offsets in workspace, thus reducing the size of the instructions needed to access them, and hence increasing the speed of execution.

Dead-code elimination. Code that cannot be reached during the execution of the program is removed.

Peephole optimisation. Code sequences are selected that are the fastest for the operation. For example, single precision floating variables are moved using the integer move operations.

Instruction scheduling Where possible the compiler exploits the internal concurrency of the transputer. In particular integer and floating point operations can be overlapped to exploit the parallel execution of the integer processor and floating point processor on the T800 series.

Constant caching. Some constants have their load time reduced by placing them in a constant table.

Unnecessary jumps are eliminated.

Switch statements can generate a number of different code sequences to cover the dense ranges within the total range.

Special idioms that are better on transputers are chosen for some code sequences.

Libraries

The full set of ANSI libraries is provided for all processor types.

The standard library operates in **double** precision. Versions of the mathematical functions are provided that operate on **float** arguments and return **float** values. These libraries provide improved performance for applications where performance requirements override accuracy requirements.

The standard C mathematical functions provided use the same code as in the OCCaM compiler. This ensures identical results and accuracy for all compilers supported by INMOS.

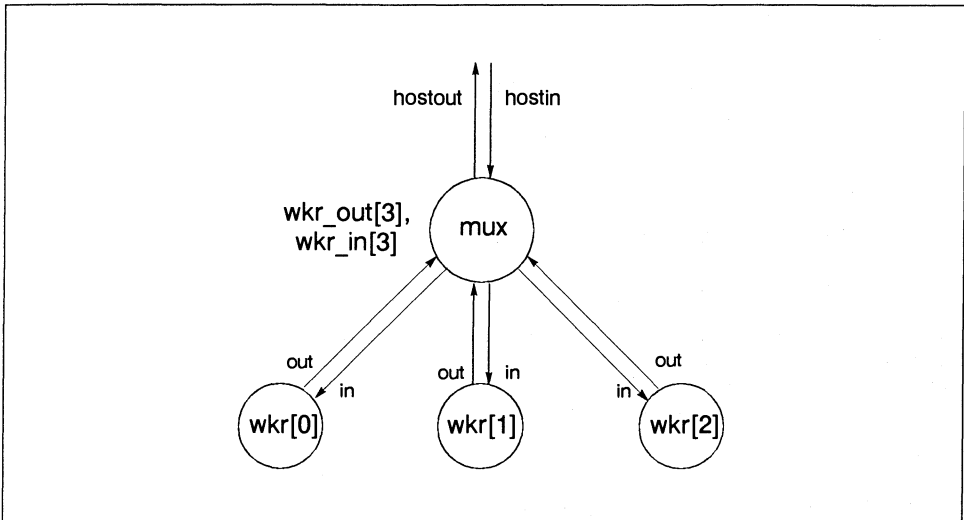


Figure 3.1 Software Network

A reduced C library is supplied to minimize code size for embedded systems applications. This library is appropriate for processes which do not need to access host operating system facilities.

Collections of functions can be compiled separately with the INMOS C compiler and then combined into a library. The linker is used to combine separately compiled functions into a program to run on a single processor. The linker supports selective loading of library units.

Mixed language programs

The ANSI C toolset allows OCCAM procedures and (single valued) OCCAM functions to be called from C just like other C functions¹

The associated OCCAM toolset supports calling C functions directly from OCCAM. Functions which require access to static variables are supported.

OCCAM and C processes may be freely mixed when configuring a program for a single transputer or a network of transputers.

Assembler inserts

The ANSI C toolset provides a very powerful assembler insert facility. The assembler insert facility supports

- Access to full instruction set of the transputer
- Symbolic access to C variables (automatic and static)
- Pseudo operations to load multiword values into registers
- Loading results of C expressions to registers using the pseudo operations.
- Labels and jumps
- Directives for instruction sizing, stack access, return address access etc.

1. Pragmas are provided to tell the compiler not to generate the hidden static link parameter (required by C but not by occam), and to change the external name of a function (occam procedures and functions may have names which are not legal C function names)


```

/* Example program */

#include <process.h>
#include <misc.h>
#include <stdlib.h>

/* Define prototypes for process functions */

void fmux (Process*, Channel*, Channel*, Channel *[], Channel *[], int);
/* process, hostin, hostout, in, out, no_workers */

void fwkr (Process*, Channel*, Channel*, int);
/* process, in, out, worker_id */

/* Main program */

int main (int argc, char *argv[], char *envp[],
          Channel *in[], int inlen, Channel *out[], int outlen)
{
    int i;

    /* Declare processes and channels */
    Channel *hostin, *hostout, *wkr_in[3], *wkr_out[3];
    Process *mux, *wkr[3];

    /* Allocate channels */
    hostin = in[1];
    hostout = out[1];
    for (i = 0; i < 3; ++i) {
        wkr_in[i] = ChanAlloc ();
        wkr_out[i] = ChanAlloc ();
    }

    /* Allocate processes */
    mux = ProcAlloc (fmux, 0, 5, hostin, hostout, wkr_in, wkr_out, 3);
    for (i = 0; i < 3; ++i) {
        wkr[i] = ProcAlloc (fwkr, 0, 3, wkr_in[i], wkr_out[i], i);
    }

    /* Start the processes running in parallel */
    ProcPar (mux, wkr[0], wkr[1], wkr[2], NULL);

    exit (0);
}

```

Figure 3.2 Parallel Processes in C

3.2.3 Target systems

The compiler will generate code for the full range of transputers; IMS M212, T212, T222, T225, T400, T414, T425, T800, T801, and T805.

The processor target type and other compilation options are specified by command line switches. Libraries may contain code compiled for several different target processors. The linker will select the correct unit at link time. The compiler, linker and librarian additionally support code compiled to run on a range of processor types achieving a space saving in libraries.

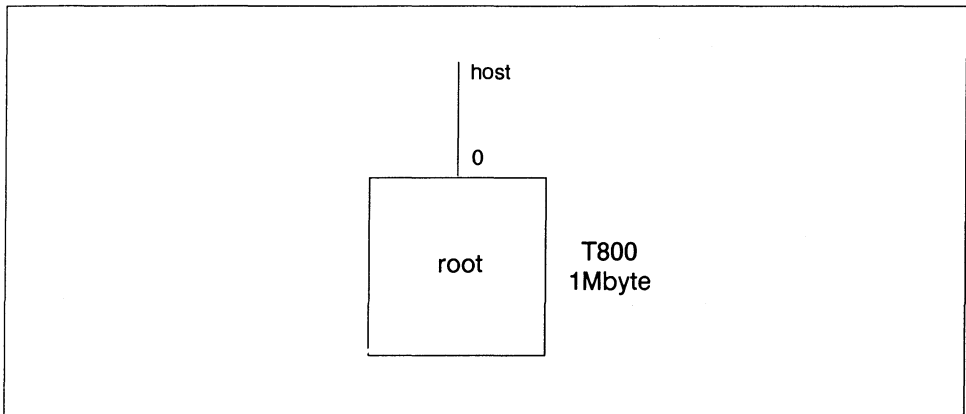


Figure 3.3 Hardware Network 1

Mixed networks may consist of any combination of any processor types. The configuration tools, eprom support tools and interactive and post-mortem debugging tools all support mixed networks.

3.2.4 Support for parallelism

The ANSI C toolset supports parallelism on individual transputers, and parallelism across networks of transputers.

By using library calls parallel processes may be created dynamically. Processes may be created individually in which case the function call will return immediately with the called process executing concurrently with the calling process, or created as a group, in which case the function call return will indicate completion of all processes within the group.

Processes have their own stack space (typically allocated from the heap of the main process) but share static and heap space. Processes created in this way can communicate by message passing over channels or by shared data protected by semaphores. Processes may be created at high and low priority levels. Interrupt routines are typically implemented as high priority processes. Functions are provided to read a message from one of a list of channels (implementing the occam `ALT` construct), to timeout on channel input, and to access the high and lower priority timers built into the transputer. The microcoded transputer scheduler provides extremely efficient scheduling of these processes.

The example in Figure 3.2 illustrates how to program the collection of parallel processes shown diagrammatically in Figure 3.1. The functions `fmux` and `fwkr` contain the executable code of the processes. These processes will communicate using message passing functions.

The linker can produce processes in the form of fully linked processes. These processes can be distributed over a network of processors using the configuration tools. The processes may communicate by message passing over channels.

A configuration language is used to describe the transputer network, the network of processes and interconnections, and the mapping of the processes onto the transputer network. Multiple processes may be mapped onto the same transputer. Communicating processes must reside on the same or adjacent processors, and only one pair of channels may be placed on a transputer link.

Two examples illustrate just how easy it is to configure programs for transputers. In both cases we assume the `mux` and `wkr` processes in the software network have been compiled and linked into file `mux.1ku` and `wkr.1ku` respectively.

Figure 3.4 shows the configuration text for mapping the software network in Figure 3.1 onto the hardware shown in Figure 3.3: a single T800 with 1 Mbyte of memory, connected to the host by link 0. In this example all the processes run on the root transputer.

```

/* Configuration example 1 */

/* Hardware description */

T800 (memory = 1M) root;
connect root.link[0] to host; /* Host is pre-defined edge */

/* Software description */

/* Define process memory sizes and interfaces */
process (stacksize = 2K, heapsize = 16k); /* Define defaults */
rep i = 0 for 3
    process (interface (input in, output out, int id)) wkr[i];
process (interface (input hostin, output hostout,
                    input in[3], output out[3])) mux;

/* Define external channels, interconnections and parameters */
input hostinput; /* Host channel edges */
output hostoutput;
connect mux.hostin to hostinput; /* Host channel connections */
connect mux.hostout to hostoutput;
rep i = 0 for 3
    {
        wkr[i] (id = i); /* Set worker process id parameter*/
        connect mux.in[i] to wkr[i].out;
        connect mux.out[i] to wkr[i].in;
    }

/* Mapping description */

/* Define linked file units for processes */
use "mux.lku" for mux;
rep i = 0 for 3
    use "wkr.lku" for wkr[i];

/* Map processes to processors and external channels to edges */
rep i = 0 for 3
    place wkr[i] on root;
place mux on root;
place hostinput on host;
place hostoutput on host;

```

Figure 3.4 Software Configuration 1

Figure 3.6 shows the configuration text for mapping the software network on to hardware shown in Figure 3.5; four T800s with 1 Mbyte of memory. In this example the `mux` process runs on the root transputer while the individual `wkr` processes run on each of the node transputers.

Only four lines of the configuration examples are different, the description of the software network is the same in both cases. Regardless of the target configuration it will always be possible to reconfigure the application for a single transputer, providing a useful first stage for target debugging.

The configuration tools can create a multi-processor program from this configuration description and the linked process units. Bootstraps and program distribution code is automatically added resulting in a program which will distribute itself across a transputer network with no additional programming required by

the user. Multi-processor programs can be loaded from a host machine using the **iserver** program, or can be converted into a range of industry standard EPROM file formats for programming into EPROM.

The interactive and post-mortem symbolic debugging tools support both forms of parallelism; parallel library and configuration level.

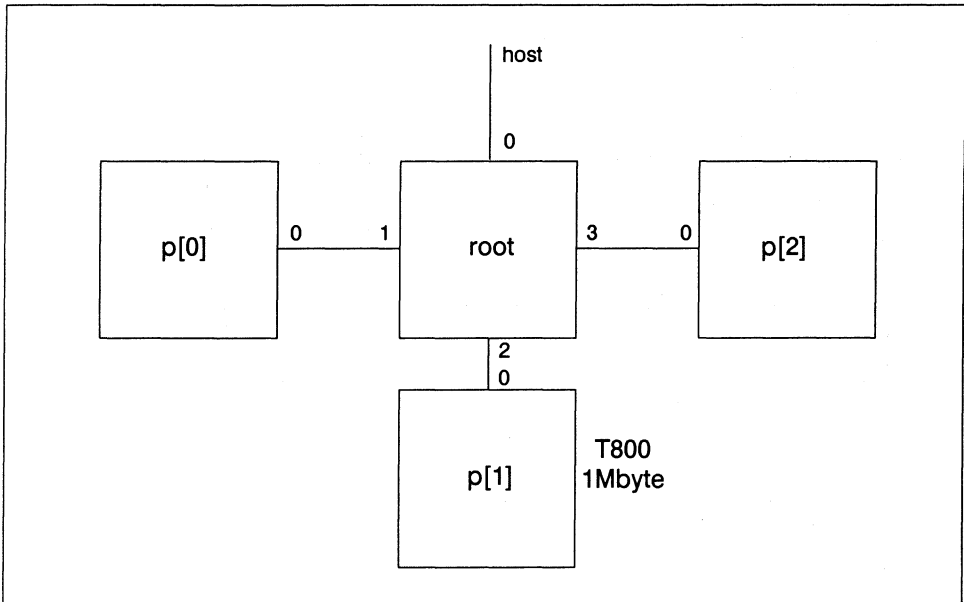


Figure 3.5 Hardware Network 2

3.2.5 Debugging

The ANSI C toolset provides three debugging tools: a T425 simulator, an interactive symbolic debugger, and a post-mortem symbolic debugger. The simulator provides debugging functions on the host machine while the interactive and post-mortem debuggers provide debugging on transputer target machines.

T425 simulation

The T425 simulator is a machine level simulation of the T425 processor connected to a host running the **iserver**. It allows transputer code to be executed on the host machine (except in the case of the IBM PC toolset which uses a transputer board to run the simulator).

The simulator provides machine level debugging support including: breakpoints and single stepping at machine code level, browsing memory in different forms including disassembled machine code, access to registers and processor queues. As explained previously, code for an arbitrary transputer network can always be configured for a single processor allowing the simulator to be used for multi-processor programs in addition to single processor programs.

A batch mode is provided for running test suites.

Interactive symbolic debugging

The interactive debugger provides source level interactive debugging across a mixed network of processors. The debugger supports breakpoints at the C or OCCAM source code level. Breakpoints may be set on any processor in the network. The state of any halted process in the network can be examined symbolically. Variables may be read or written symbolically, C expressions may be evaluated allowing

access to **struct** fields and pointer chasing. The stack can be backtraced to examine the nesting of function calls; it is even possible to backtrace through the calls which create parallel threads to the parent processes. Channel variables can be inspected. If another process is waiting on a channel it is possible to jump through the channel and inspect the state of the waiting process, even if it is on another processor. The debugger will automatically switch between OCCAM and C when debugging a mixed language program.

The interactive debugger provides all the machine level facilities of the simulator for every processor in the network (with the exception of machine level single stepping and register modification).

```

/* Configuration example 2 */

/* Hardware description */

T800 (memory = 1M) root, p[3];
connect root.link[0] to host; /* Host link connection */
rep i = 0 for 3
    connect root.link[i + 1] to p[i].link[0];

/* Software description */

/* Define process memory sizes and interfaces */
process (stacksize = 2K, heapsize = 16k); /* Define defaults */
rep i = 0 for 3
    process (interface (input in, output out, int id)) wkr[i];
process (interface (input hostin, output hostout,
                    input in[3], output out[3])) mux;

/* Define external channels, interconnections and parameters */
input hostinput; /* Host channel edges */
output hostoutput;
connect mux.hostin to hostinput; /* Host channel connections */
connect mux.hostout to hostoutput;
rep i = 0 for 3
{
    wkr[i] (id = i); /* Set worker process id parameter*/
    connect mux.in[i] to wkr[i].out;
    connect mux.out[i] to wkr[i].in;
}

/* Mapping description */

/* Define linked file units for processes */
use "mux.lku" for mux;
rep i = 0 for 3
    use "wkr.lku" for wkr[i];

/* Map processes to processors and external channels to edges */
rep i = 0 for 3
    place wkr[i] on p[i];
place mux on root;
place hostinput on host;
place hostoutput on host;

```

Figure 3.6 Software Configuration 2

Post-mortem symbolic debugging

The post-mortem debugger can be used to examine the state of a transputer network symbolically. The debugger works with exactly the same code as will run in your final product; there is no additional code inserted to support debugging. This supports the cases where the program works under simulation, works when debugging is compiled in, but fails when the debugging is removed. After a program has halted or been interrupted by the developer, the state of the network can be preserved so that the post-mortem debugger can be run. The post-mortem debugger will support direct analysis of the network, or allow the state of the network to be saved in a dump file for later analysis. The post-mortem debugger supports the same symbolic and machine level browsing functions as the interactive debugger.

Both the interactive debugger and the post-mortem debugger require a transputer to run. The simulator will run directly on Sun 3, Sun 4 and VAX hosts.

3.2.6 Improvements over previous releases

The D7214, D6214, D5214, and D4214 ANSI C toolsets represent a considerable improvement over the D711, D611 and D511 Parallel C compilers. A summary is provided below.

- ANSI C rather than K & R C.
- Faster code generated.
- Tools execute native on Sun and VAX hosts.
- Faster execution of compiler.
- New debugging tools: including interactive symbolic debugger with breakpoint support, improved post-mortem debugging, and transputer simulation on the host machine.
- Configuration language updated ready to support next generation transputers.
- Improved parallel library support.
- Improved mixing of C and OCCam.
- Improved assembler insert support.
- EPROM support included.
- Makefile generator included.

With the longer term in mind the object file format for this release has been improved. Conversion tools are provided to convert old format object files into the new format. Given the performance improvements offered, it is recommended that where possible existing code should be recompiled into the new format. The parallel library has been completely redesigned and now correctly supports the OCCam PAR and ALT constructs, and semaphores now work across different process priorities. The old thread library has been included to ease transition to the new structure. This will not be supported in future releases of the software.

3.3 ANSI C Toolset Product Components

3.3.1 Documentation

- Delivery manual
- User manual
- Reference manual
- ANSI C toolset handbook

3.3.2 Software Tools

icc, llink, llbr – ANSI C compiler, linker and librarian
lmakef, llst – Makefile generator and binary lister program
lconf, lcollect – configuration tools
lsm, ldebug, lskip, ldump – debugging tools
lserver – INMOS host server program
leprom, lemit – eprom and memory interface programming tools
lcvmlink, lemivt – conversion tools for old file formats

3.3.3 Software libraries

llibc.lib – Full ANSI library plus parallel support
llibcred.lib – Reduced library for embedded systems
lcentry.lib – Mixed language support library

3.4 Product Variants

3.4.1 IMS D7214 IBM PC version

Although the PC tools are invoked as if they were ordinary PC resident tools, they actually run on a transputer board plugged into the PC. A number of the tools are additionally provided in a form which will run directly on the PC.

Operating requirements

You will need one of the following configurations:

- IBM PC XT or AT with 512Kbyte memory, an IMS B008 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3)
- NEC PC-9801 with 512Kbyte memory, an IMS B015 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3).

In each case you will require:

- DOS 3.0 or later
- 7 Mbyte of free disk space

The interactive symbolic debugger requires an additional 2 Mbyte IMS B404 TRAM. The simulator and symbolic post-mortem debugger do not require this additional TRAM.

Distribution media

Software is distributed on two media systems:

- 360 Kbyte (48TP) 5.25 inch IBM format floppy disks
- 720 Kbyte 3.5 inch IBM format floppy disks.

3.4.2 IMS D6214 VAX VMS version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Operating requirements

For hosted cross-development you will need:

- VAX VMS 5.0 or later
- 10 Mbytes of free disk space.

For loading target systems and target debugging you will need one of the following:

- A third party interface board supporting a connection to a 32 bit transputer with 2 Mbytes of memory
- An IBM PC cross development system, plus DECNET connection.

Distribution media

Software is distributed on a TK50 tape cartridge in VMS backup format.

3.4.3 IMS D5214 Sun 3 version, IMS D4214 Sun 4 version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Operating requirements

For hosted cross-development and debugging you will require:

- A Sun 3 or Sun 4 workstation or server
- SunOS 4.0.3 or later
- 10 Mbytes of free disk space.

For loading target systems and remote debugging you will require one of the following:

- IBM PC development system plus PC-NFS.

The interactive symbolic debugger requires an additional 2 Mbyte IMS B404 TRAM. The simulator and symbolic post-mortem debugger do not require this additional TRAM.

A very high performance interface to up to 4 transputer networks can be constructed using the IMS B016 VME master board.

Distribution media

Sun 3 software is distributed on DC600A data cartridges 60 Mbyte, QIC-11, tar format. Sun 4 software is distributed on DC600A data cartridges 60 Mbyte, QIC-24, tar format.

3.5 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates.

Software problem report forms are included with the software.

INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.

glockenspiel
class constructors

IMS D7217
IMS D6217
IMS D5217
IMS D4217

C + +

Product Information



C + + native and transputer cross-development systems for IBM PC, NEC PC, Sun 3, Sun 4 and VAX hosts

KEY FEATURES

- Conformance with the C + + 2.0 specification
- Glockenspiel C + + cross compiler targeting INMOS ANSI C compiler
- Native Glockenspiel C + + targeting host C compiler
- Support for parallelism via INMOS ANSI C libraries
- Support for all INMOS processors (32 and 16 bit)
- Interactive symbolic debugging for transputer networks
- Post-mortem symbolic debugging for transputer networks
- Support for EPROM programming
- Consistent tools across PC, VAX VMS, Sun 3 and Sun 4 hosts

APPLICATIONS

- Embedded systems (both single and multiple transputers)
- Porting of existing software and packages
- Evaluation of transputers for concurrent applications
- Scientific programming

4.1 Product Overview

Glockenspiel were the first company to license C++ from AT&T in 1985. Since then they have been developing and supporting C++ compilers and libraries.

The toolset includes a native C++ compiler and a C++ cross compiler which when used in conjunction with the INMOS ANSI C toolset can be used to program transputer networks. The product is distributed and supported by INMOS.

The INMOS C++ Toolset is available for four development platforms:

IMS D7217 C++ Toolset for IBM and NEC PC under PCDOS 3.0 or later

IMS D6217 C++ Toolset for VAX under VMS 5.0 or later

IMS D5217 C++ Toolset for Sun 3 under SunOS 4.0.3 or later

IMS D4217 C++ Toolset for Sun 4 under SunOS 4.0.3 or later

4.1.1 C++

C++ is a general purpose programming language which has evolved from C. It combines the benefits of object orientated programming with the efficiency of C.

Its benefits include:

- **Strong Type Checking**
Help reduce coding problems
- **Encapsulation**
Enable large applications by using C++ classes. A C++ class is a user defined type which may contain data members to represent the type and function members to implement operations on the type.
- **Data Abstraction**
Ease maintenance and product evolution by restricting access to a class's implementation details.
- **Multiple Inheritance**
Classes may inherit properties from other classes. This enables classes (and hence effort) to be reused.
- **Dynamic Binding**
Use function names consistently, independent of object type, as class members may be bound dynamically at run-time (virtual functions).
- **Type-Safe Linkage**
Provides function argument checking across different compilation modules. This enables the correct function to be acquired by a linker when several alternatives are available in the presence of function overloading.

4.1.2 Use with the INMOS ANSI C toolset

The following capabilities of the INMOS ANSI C toolset can be used from C++

- **Transputer support library**
Functions for creating parallel processes and communicating between them

- Multiprocessor configuration tools
Building programs to run on networks of transputers.

4.2 Glockenspiel C++ Product Components

4.2.1 Documentation

- Installation Guide
- Comprehensive User manual
- Single A4 page quick reference guide
- Programming in C++ by Steve Dewhurst and Kathy Stark

4.2.2 Software tools

The tools have been named such that the cross-development tools share the same name as the corresponding native tool prefixed with the letter I (eg. **ccxx** is the native compilation driver and **iccxx** is the cross-development compilation driver). This enables the switch between native and cross-development environments to be almost transparent.

In normal usage only the C++ driver will be called directly by a user.

ccxx, iccxx	C++ compilation driver
gcpp, igcpp	C++ preprocessor
cfx, icfx	C++ compiler
mxx, imxx	C++ constructor linker
fx, ifx	C++ debug information filter

4.2.3 Software Libraries

libcxx.lib	C++ iostream and C++ run-time support library
libcplx.lib	C++ complex math class library

4.3 Product Variants

4.3.1 IMS D7217 IBM PC version

Although the PC cross-development tools are invoked as if they were ordinary PC resident tools, they actually run on the transputer board plugged into the PC. A number of the tools are additionally provided in a form which will run directly on the PC.

Cross-development operating requirements

For hosted cross-development you will need one of the following configurations

- IBM PC AT with 512Kbyte memory, an IMS B008 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3)
- NEC PC-9801 with 512Kbyte memory, an IMS B015 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3)

In each case you will require

- DOS 3.0 or later

- IMS D7214 ANSI C Toolset (requires 7 Mbytes of free disk space)
- 3 Mbyte of free disk space

For interactive transputer symbolic debugging an additional 2Mbyte IMS B404 TRAM is required.

Native development operating requirements

For native development you will need

- 80286 PC with 640Kbyte memory and at least 1Mbyte of extended memory.
- DOS 3.0 or later
- 3 Mbyte of free disk space
- Microsoft C 6.0

Distribution media

Software is distributed on **BOTH** 1.2 Mbyte (96TPI) 5.25 inch IBM format floppy disks **AND** 1.44 Mbyte 3.5 inch IBM format floppy disks.

4.3.2 IMS D6217 VAX VMS version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Cross-development operating requirements

For hosted cross-development you will need

- VAX VMS 5.0 or later
- IMS D6214 ANSI C Toolset (requires 10 Mbytes of free disk space)
- 4 Mbytes of free disk space

For loading target systems you will require one of the following:

- A third party interface board supporting a connection to a 32 bit transputer with 2 Mbytes of memory.
- IBM PC development system plus DECNET connection

For interactive transputer symbolic debugging an additional 2Mbyte IMS B404 TRAM is required.

Native development operating requirements

For native development you will need

- VAX VMS 5.0 or later
- VAX VMS C
- 4 Mbytes of free disk space

Distribution media

Software is distributed on a TK50 tape cartridge in VMS backup format.

Licensing variants

There are different licensing variants of the Glockenspiel C++ product depending on the machine on which you wish to use the product (shown in table 4.1). Each product provides a licence for use on a single machine of the designated model type.

Computer	Models	Variant
DEC MicroVAX	11,2***,3***	IMS D6217-B
	33**,34**	IMS D6217-C
	35**,36**,38**	IMS D6217-D
	39**	IMS D6217-E
DEC VAX	730	IMS D6217-B
	750,780	IMS D6217-C
	8250,8350,6210	IMS D6217-D
	8800,8810,85**,86**,87**	IMS D6217-E
	6220,6230,6310,6320,6410	IMS D6217-E
	8820,6240,6330,6420	IMS D6217-F
	8830,6340,6350,6430	IMS D6217-G
	8840,89**,6360,6440,6450,6460	IMS D6217-H
	9000	IMS D6217-H
VAXstation	11,2***,3***	IMS D6217-WA
	3***	IMS D6217-WB

Table 4.1 IMS D6214 Licensing variants

4.3.3 IMS D5217 Sun 3 version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Cross-development operating requirements

For hosted cross-development you will require:

- A Sun 3 workstation or server
- SunOS 4.0.3 or later
- IMS D5214 ANSI C Toolset (requires 10 Mbytes of free disk space)
- 3 Mbytes of free disk space

For loading target systems you will require one of the following:

- IMS B014 VME motherboard with 2 Mbyte IMS B404 TRAM (eg. IMS B404-3)
- IBM PC development system plus PC-NFS

For interactive transputer symbolic debugging an additional 2Mbyte IMS B404 TRAM is required.

A very high performance interface to up to 4 transputer networks can be constructed using the IMS B016 VME master board.

Native development operating requirements

For native development you will need:

- A Sun 3 workstation or server
- SunOS 4.0.3 or later
- 3 Mbytes of free disk space

Distribution media

Sun 3 software is distributed on DC 600A data cartridges, QIC-11, tar format.

Licensing variants

There is only one licence option which provides a single machine licence for any Sun 3 machine regardless of the model.

4.3.4 IMS D4217 Sun 4 version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Cross-development operating requirements

For hosted cross-development you will require

- Sun 4 workstation or server
- SunOS 4.0.3 or later
- IMS D4214 ANSI C Toolset (requires 10 Mbytes of free disk space)
- 3 Mbytes of free disk space

For loading target systems you will require one of the following:

- IMS B014 VME motherboard with 2 Mbyte IMS B404 TRAM (eg. IMS B404-3)
- IBM PC development system plus PC-NFS

For interactive transputer symbolic debugging an additional 2Mbyte IMS B404 TRAM is required.

A very high performance interface to up to 4 transputer networks can be constructed using the IMS B016 VME master board.

Native development operating requirements

For native development you will need

- A Sun 4 workstation or server
- SunOS 4.0.3 or later
- 3 Mbytes of free disk space

Licensing variants

There are different licensing variants of the Glockenspiel C++ product depending on the machine on which you wish to use the product (shown in table 4.2). Each product provides a licence for use on a single machine of the designated model type.

Computer	Models	Variant
Sun 4 workstation	20ws, 60ws	IMS D4217-WA
	40ws, 65ws, 75ws, 1**ws, 330ws	IMS D4217-WB
	2**ws,370ws, 470ws	IMS D4217-WC
Sun 4 server	60s	IMS D4217-WB
	65s, 75s, 330s	IMS D4217-WC
	2**s,370s,470s	IMS D4217-WD
	390s,490s	IMS D4217-WE

Table 4.2 IMS D4217 Licensing variants

Distribution media

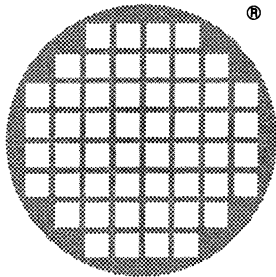
Sun 4 software is distributed on DC 600A data cartridges, QIC-24, tar format.

4.4 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates.

Software problem report forms are included with the software.

INMOS has a world-wide network of sales offices, providing support for INMOS products. In some areas the support functions may be taken over by distributors or other third parties.



inmos®

IMS D7216
IMS D6216
IMS D5216
IMS D4216

ANSI FORTRAN 77 Toolset

Advance Information

ANSI FORTRAN 77 cross-development systems for IBM PC, NEC PC, Sun 3, Sun 4 and VAX hosts

KEY FEATURES

- Full ANSI FORTRAN 77 (ANSI X3.9 -1978)
- Validated against FSVC78 version 2.0 from FSTC
- Excellent compile time diagnostics
- Optimised code generation
- Support for parallelism
- Interactive symbolic debugging for transputer networks
- Post-mortem symbolic debugging for transputer networks
- T425 simulator
- Consistent tools across PC, VAX VMS, Sun 3, and Sun 4 hosts

APPLICATIONS

- Scientific and Numeric programming for accelerators
- Porting of existing software and packages
- Evaluation of transputers for concurrent applications
- Embedded systems (both single and multiple transputers)

5.1 Introduction

The following discussion outlines the use and features of the ANSI FORTRAN toolset in moderate technical detail. In particular examples are included to illustrate how easily parallel programs and multi-processor programs can be built using the ANSI FORTRAN toolset. A summary of the documentation and tools provided in the toolset is given. Finally the operating requirements and distribution media for each of the PC, VAX and Sun products are specified.

5.2 Product Overview

The INMOS FORTRAN Toolsets provide complete FORTRAN cross-development systems for transputer targets. They can be used to build parallel programs for single transputers and for multi-transputer networks consisting of arbitrary mixtures of transputer types. Programs developed with the toolset are both source and binary compatible across all host development machines. The INMOS ANSI FORTRAN 77 Toolset is available for four development platforms:

IMS D7216 ANSI FORTRAN 77 Toolset for IBM and NEC PC under PCDOS

IMS D6216 ANSI FORTRAN 77 Toolset for VAX VMS

IMS D5216 ANSI FORTRAN 77 Toolset for Sun 3 under SunOS 4.0.3

IMS D4216 ANSI FORTRAN 77 Toolset for Sun 4 under SunOS 4.0.3

5.2.1 How programs are built

FORTRAN 77 programs may be separately compiled. Separately compiled units may optionally be collocated into libraries. The linker links separately compiled units and libraries into fully resolved main program units. Collections of communicating main program units may be distributed across networks of transputers using the configuration tools. The results of configuration may be loaded down a transputer link using the host server. In addition to loading programs down a transputer link, the host server program provides access to host operating system facilities through a remote procedure call mechanism. The server program is provided in FORTRAN source code to allow extension by developers. For developers using a make program, a tool **lmakef** is provided which will generate a Makefile automatically. This automates the build process and guarantees that the Makefile is consistent with program sources.

5.2.2 ANSI FORTRAN compilation system

Compiler operation

The compiler operates from a host command line interface. The compile time diagnostics provided by the compiler are truly excellent.

ANSI conformance

The INMOS ANSI FORTRAN 77 Toolset supports the full standard language as defined in X3.9-1978 plus a number of VAX and IBM FORTRAN extensions.

Optimised code generation

The ANSI FORTRAN 77 toolset achieves over 20 percent improvement in program execution speed over previous INMOS FORTRAN compilers as measured by an internal set of program benchmarks. The compiler implements a wide range of code optimisation techniques.

Constant folding. The compiler evaluates all integer and real constant expressions at compile time.

Workspace allocation. Frequently used variables are placed at small offsets in workspace, thus reducing the size of the instructions needed to access them, and hence increasing the speed of execution.

Dead-code elimination. Code that cannot be reached during the execution of the program is removed.

Peephole optimisation. Code sequences are selected that are the fastest for the operation. For example, single precision floating variables are moved using the integer move operations.

Instruction scheduling Where possible the compiler exploits the internal concurrency of the transputer. In particular integer and floating point operations can be overlapped to exploit the parallel execution of the integer processor and floating point processor on the T800 series.

Constant caching. Some constants have their load time reduced by placing them in a constant table.

Unnecessary jumps are eliminated.

Special idioms that are better on transputers are chosen for some code sequences.

Run-Time System

The standard FORTRAN mathematical functions provided use the same code as in the OCCaM compiler. This ensures identical results and accuracy for all compilers supported by INMOS.

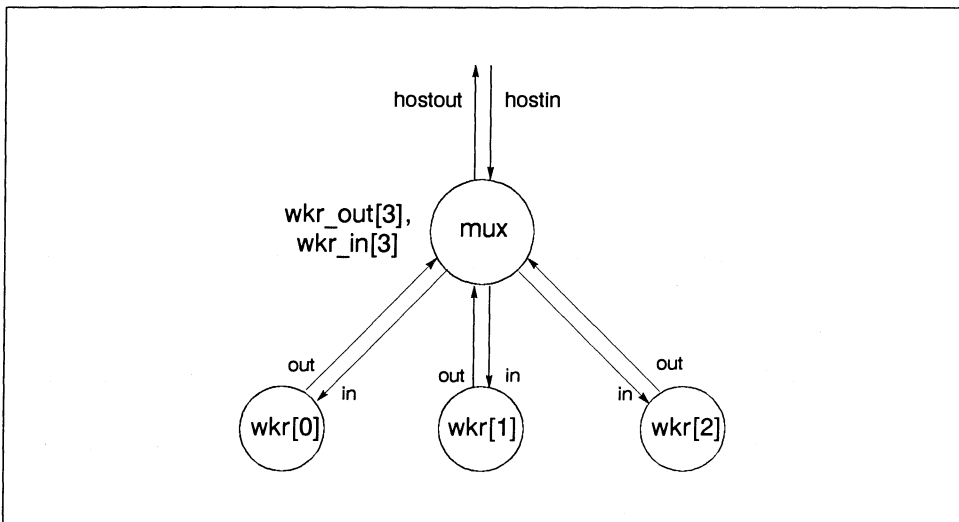


Figure 5.1 Software Network

A reduced FORTRAN 77 run-time system is supplied to minimize code size for embedded systems applications. This library is appropriate for processes which do not need to access host operating system facilities.

Collections of functions can be compiled separately with the INMOS FORTRAN 77 compiler and then combined into a library. The linker is used to combine separately compiled functions into a program to run on a single processor. The linker supports selective loading of library units.

Mixed language programs

The ANSI FORTRAN 77 toolset allows C functions, OCCaM procedures and (single valued) OCCaM functions to be called from FORTRAN.

The associated OCCaM toolset supports calling FORTRAN functions directly from OCCaM.

FORTRAN, OCCaM and C processes may be freely mixed when configuring a program for a single transputer or a network of transputers.

5.2.3 Target systems

The compiler will generate code for the following transputers; T400, T414, T425, T800, T801, and T805.

The processor target type and other compilation options are specified by command line switches. Libraries may contain code compiled for several different target processors. The linker will select the correct unit at link time. The compiler, linker and librarian additionally support code compiled to run on a range of processor types achieving a space saving in libraries.

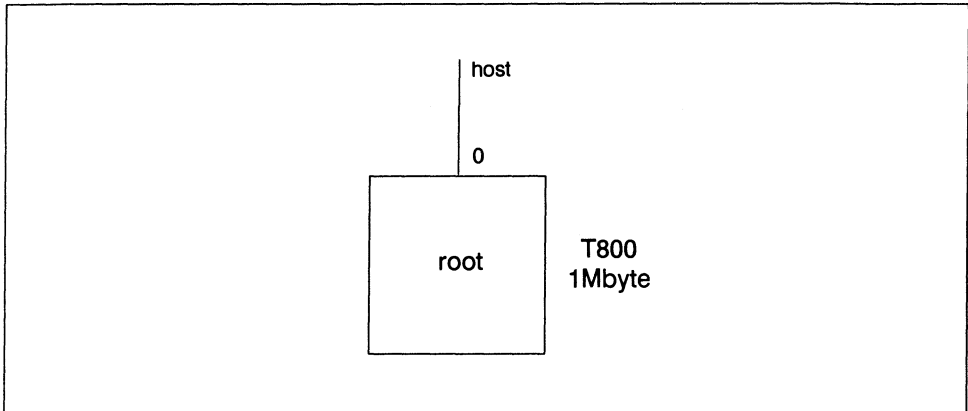


Figure 5.2 Hardware Network 1

Mixed networks may consist of any combination of any processor types. The configuration tools and interactive and post-mortem debugging tools all support mixed networks.

5.2.4 Support for parallelism

The ANSI FORTRAN 77 toolset supports parallelism on individual transputers, and parallelism across networks of transputers.

The tools support programming of farming, pipelines and data parallelisms over arrays of processors.

Processes may be created at high and low priority levels. Interrupt routines are typically implemented as high priority processes. Subroutines are provided to read a message from one of a list of channels (implementing the occam `ALT` construct), to timeout on channel input, and to access the high and lower priority timers built into the transputer. The microcoded transputer scheduler provides extremely efficient scheduling of these processes.

The linker can produce processes in the form of fully linked processes. These processes can be distributed over a network of processors using the configuration tools. The processes may communicate by message passing over channels.

A configuration language is used to describe the transputer network, the network of processes and interconnections, and the mapping of the processes onto the transputer network. Multiple processes may be mapped onto the same transputer. Communicating processes must reside on the same or adjacent processors, and only one pair of channels may be placed on a transputer link.

Two examples illustrate just how easy it is to configure programs for transputers. In both cases we assume the `mux` and `wkr` processes in the software network have been compiled and linked into file `mux.lku` and `wkr.lku` respectively.

Figure 3.4 shows the configuration text for mapping the software network in Figure 3.1 onto the hardware shown in Figure 3.3: a single T800 with 1 Mbyte of memory, connected to the host by link 0. In this example all the processes run on the root transputer.

```

/* Configuration example 1 */

/* Hardware description */

T800 (memory = 1M) root;
connect root.link[0] to host; /* Host is pre-defined edge */

/* Software description */

/* Define process memory sizes and interfaces */
process (stacksize = 2K, heapsize = 16k); /* Define defaults */
rep i = 0 for 3
  process (interface (input in, output out, int id)) wkr[i];
process (interface (input hostin, output hostout,
                  input in[3], output out[3])) mux;

/* Define external channels, interconnections and parameters */
input hostinput; /* Host channel edges */
output hostoutput;
connect mux.hostin to hostinput; /* Host channel connections */
connect mux.hostout to hostoutput;
rep i = 0 for 3
  {
    wkr[i] (id = i); /* Set worker process id parameter*/
    connect mux.in[i] to wkr[i].out;
    connect mux.out[i] to wkr[i].in;
  }

/* Mapping description */

/* Define linked file units for processes */
use "mux.lku" for mux;
rep i = 0 for 3
  use "wkr.lku" for wkr[i];

/* Map processes to processors and external channels to edges */
rep i = 0 for 3
  place wkr[i] on root;
place mux on root;
place hostinput on host;
place hostoutput on host;

```

Figure 5.3 Software Configuration 1

Figure 3.6 shows the configuration text for mapping the software network on to hardware shown in Figure 3.5; four T800s with 1 Mbyte of memory. In this example the `mux` process runs on the root transputer while the individual `wkr` processes run on each of the node transputers.

Only four lines of the configuration examples are different, the description of the software network is the same in both cases. Regardless of the target configuration it will always be possible to reconfigure the application for a single transputer, providing a useful first stage for target debugging.

The configuration tools can create a multi-processor program from this configuration description and the linked process units. Bootstraps and program distribution code is automatically added resulting in a program which will distribute itself across a transputer network with no additional programming required by the user. Multi-processor programs can be loaded from a host machine using the `lserver` program.

The interactive and post-mortem symbolic debugging tools support this parallelism.

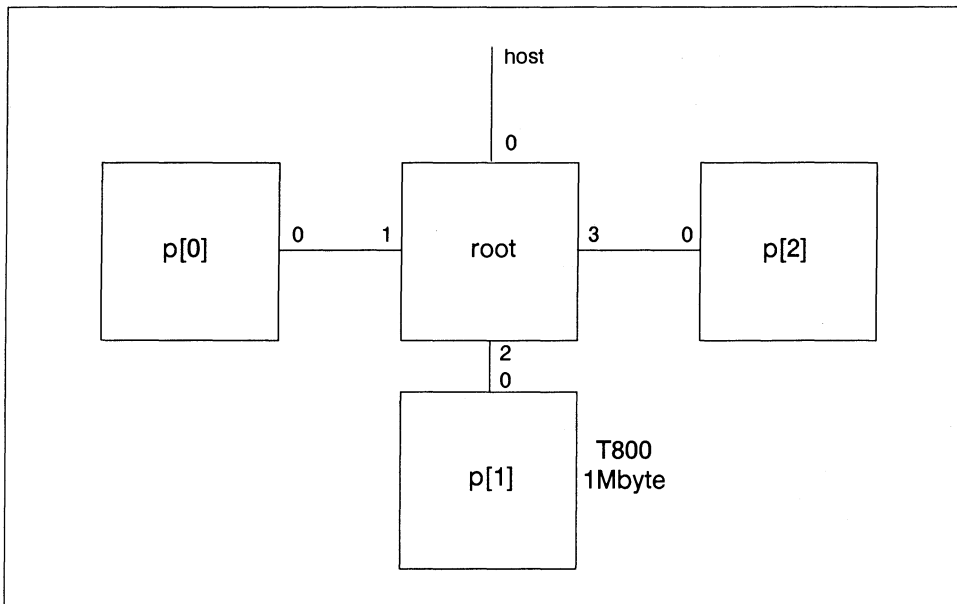


Figure 5.4 Hardware Network 2

5.2.5 Debugging

The F77 toolset provides three debugging tools: a T425 simulator, an interactive symbolic debugger, and a post-mortem symbolic debugger. The simulator provides debugging functions on the host machine while the interactive and post-mortem debuggers provide debugging on transputer target machines.

T425 simulation

The T425 simulator is a machine level simulation of the T425 processor connected to a host running the **iserver**. It allows transputer code to be executed on the host machine (except in the case of the IBM PC toolset which uses a transputer board to run the simulator).

The simulator provides machine level debugging support including: breakpoints and single stepping at machine code level, browsing memory in different forms including disassembled machine code, access to registers and processor queues. As explained previously, code for an arbitrary transputer network can always be configured for a single processor allowing the simulator to be used for multi-processor programs in addition to single processor programs.

A batch mode is provided for running test suites.

Interactive symbolic debugging

The interactive debugger provides source level interactive debugging across a mixed network of processors. The debugger supports breakpoints at the FORTRAN, OCCAM or C source code level. Breakpoints may be set on any processor in the network. The state of any halted process in the network can be examined symbolically. Variables may be read or written symbolically. The stack can be backtraced to examine the nesting of subroutine calls. The debugger will automatically switch between FORTRAN, OCCAM and C when debugging a mixed language program.

The interactive debugger provides all the machine level facilities of the simulator for every processor in the network (with the exception of machine level single stepping and register modification).

```

/* Configuration example 2 */

/* Hardware description */

T800 (memory = 1M) root, p[3];
connect root.link[0] to host; /* Host link connection */
rep i = 0 for 3
  connect root.link[i + 1] to p[i].link[0];

/* Software description */

/* Define process memory sizes and interfaces */
process (stacksize = 2K, heapsize = 16k); /* Define defaults */
rep i = 0 for 3
  process (interface (input in, output out, int id)) wkr[i];
process (interface (input hostin, output hostout,
                  input in[3], output out[3])) mux;

/* Define external channels, interconnections and parameters */
input hostinput; /* Host channel edges */
output hostoutput;
connect mux.hostin to hostinput; /* Host channel connections */
connect mux.hostout to hostoutput;
rep i = 0 for 3
  {
    wkr[i] (id = i); /* Set worker process id parameter*/
    connect mux.in[i] to wkr[i].out;
    connect mux.out[i] to wkr[i].in;
  }

/* Mapping description */

/* Define linked file units for processes */
use "mux.lku" for mux;
rep i = 0 for 3
  use "wkr.lku" for wkr[i];

/* Map processes to processors and external channels to edges */
rep i = 0 for 3
  place wkr[i] on p[i];
place mux on root;
place hostinput on host;
place hostoutput on host;

```

Figure 5.5 Software Configuration 2

Post-mortem symbolic debugging

The post-mortem debugger can be used to examine the state of a transputer network symbolically. The debugger works with exactly the same code as will run in your final product; there is no additional code inserted to support debugging. This supports the cases where the program works under simulation, works when debugging is compiled in, but fails when the debugging is removed. After a program has halted or been interrupted by the developer, the state of the network can be preserved so that the post-mortem debugger can be run. The post-mortem debugger will support direct analysis of the network, or allow the state of the network to be saved in a dump file for later analysis. The post-mortem debugger supports the same symbolic and machine level browsing functions as the interactive debugger.

Both the interactive debugger and the post-mortem debugger require a transputer to run. The simulator will run directly on Sun 3, Sun 4 and VAX hosts.

5.2.6 Improvements over previous releases

The D7214, D6214, D5214, and D4214 ANSI FORTRAN toolsets represent a considerable improvement over the D711, D611 and D511 Parallel FORTRAN compilers. A summary is provided below.

- Extend language features.
- Faster code generated.
- Tools execute native on Sun and VAX hosts.
- Faster execution of compiler.
- New debugging tools: including interactive symbolic debugger with breakpoint support, improved post-mortem debugging, and transputer simulation on the host machine.
- Configuration language updated ready to support next generation transputers.
- Improved mixing of FORTRAN, C and occam.
- Makefile generator included.

With the longer term in mind the object file format for this release has been improved. Conversion tools are provided to convert old format object files into the new format. Given the performance improvements offered, it is recommended that where possible existing code should be recompiled into the new format.

5.3 ANSI FORTRAN Toolset Product Components

5.3.1 Documentation

- Delivery manual
- User manual
- Reference manual
- ANSI FORTRAN 77 toolset handbook

5.3.2 Software Tools

if77, ilink, ilibr – ANSI FORTRAN compiler, linker and librarian

imakef, ilist – Makefile generator and binary lister program

icconf, icollect – configuration tools

isim, ldebug, lskip, ldump – debugging tools

lserver – INMOS host server program

icvtlink, lemicvt – conversion tools for old file formats

5.4 Product Variants

5.4.1 IMS D7216 IBM PC version

Although the PC tools are invoked as if they were ordinary PC resident tools, they actually run on a transputer board plugged into the PC. A number of the tools are additionally provided in a form which will run directly on the PC.

Operating requirements

You will need one of the following configurations:

- IBM PC XT or AT with 512Kbyte memory, an IMS B008 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3)
- NEC PC-9801 with 512Kbyte memory, an IMS B015 Motherboard, plus a transputer module with 2 Mbytes of memory (eg. IMS B404-3).

In each case you will require:

- DOS 3.0 or later
- 7 Mbyte of free disk space

The interactive symbolic debugger requires an additional 2 Mbyte IMS B404 TRAM. The simulator and symbolic post-mortem debugger do not require this additional TRAM.

Distribution media

Software is distributed on two media systems:

- 360 Kbyte (48TPI) 5.25 inch IBM format floppy disks
- 720 Kbyte 3.5 inch IBM format floppy disks.

5.4.2 IMS D6216 VAX VMS version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Operating requirements

For hosted cross-development you will need:

- VAX VMS 5.0 or later
- 10 Mbytes of free disk space.

For loading target systems and target debugging you will need one of the following:

- A third party interface board supporting a connection to a 32 bit transputer with 2 Mbytes of memory
- An IBM PC cross development system, plus DECNET connection.

Distribution media

Software is distributed on a TK50 tape cartridge in VMS backup format.

5.4.3 IMS D5216 Sun 3 version, IMS D4214 Sun 4 version

All tools are provided in a form which will run on the host machine, and in a form which will run on transputers. The exceptions to this rule are the server, which will only run on the host machine, and the target debugging tools which will only run on a transputer.

Operating requirements

For hosted cross-development and debugging you will require:

- A Sun 3 or Sun 4 workstation or server
- SunOS 4.0.3 or later
- 10 Mbytes of free disk space.

For loading target systems and remote debugging you will require one of the following:

- IBM PC development system plus PC-NFS.

The interactive symbolic debugger requires an additional 2 Mbyte IMS B404 TRAM. The simulator and symbolic post-mortem debugger do not require this additional TRAM.

A very high performance interface to up to 4 transputer networks can be constructed using the IMS B016 VME master board.

Distribution media

Sun 3 software is distributed on DC600A data cartridges 60 Mbyte, QIC-11, tar format. Sun 4 software is distributed on DC600A data cartridges 60 Mbyte, QIC-24, tar format.

5.5 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates.

Software problem report forms are included with the software.

INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.

ALSYS

Ada Compiler



Product Information

Ada cross-development systems for IBM PC, and VAX hosts

KEY FEATURES

- Full Ada Compiler (ANSI/MIL-STD 1815A)
- Validated by United States Government Joint Ada Program Office
- Support for distributed Ada applications
- Ada specific code optimisations
- Optimised run time efficiency for multi-tasking
- Support for all 32 bit INMOS processors (T4 and T8)
- Supports calling occam from Ada.
- Access to low-level programming features
- Support for EPROM programming
- Interworks with INMOS occam 2 toolset
- Consistent with native IBM PC and VAX Ada development tools

APPLICATIONS

- Embedded systems (both single and multiple transputers)
- Porting of existing software and packages
- Evaluation of transputers for concurrent applications
- Military developments

6.1 Product Overview

Version 5 of the Alslys Ada Transputer Cross Compiler offers a complete production quality Ada environment suitable for the development real-time embedded applications. The compiler runs on all VAX, MicroVAX and VAXstations under VMS, or on an IBM PC with a transputer board. The compiler generates code for both the T800 and T400 series transputers.

The compilation system consists of: the compiler (including the high level and low level optimisers), binder, multi-library environment (family, library and unit managers), Ada run-time executive, AdaWorld (the user interface common to all Alslys compilers), optional Ada Toolset (AdaProbe, AdaXref, AdaMake and AdaReformat), standard Ada packages and user documentation.

The compiler has been officially validated by the United States Government Ada Joint Program Office.

6.2 Product Highlights

6.2.1 Supports easy implementation of distributed Ada applications

The transputer family of processors provides an excellent architecture for developing distributed multi-processor applications. Ada programs running on separate transputers can exploit rapid interprogram communication through transputer links using the CHANNEL_IO interface.

6.2.2 Efficient sharing of Ada Libraries

The Multi-Library Environment provides a powerful and flexible way to manage Ada development efforts and share program units even across local area networks. The new Version 5 Multi-Library Environment supports re-use and distribution of software components through mechanisms that copy individual compilation units and export and import entire libraries.

6.2.3 Generates high performance, compact application code

Ada Code Generation

The compiler generates extremely high quality code for fast, compact applications. A full implementation of pragma INLINE is supported. The binder supports unused subprogram elimination, removing all code for subprograms that are not called. The High Level Optimiser excels at removing constraint checks and detecting pending execution time errors at compile time. The Low Level Optimiser further reduces code size and increases speed by removing common subexpressions and passing information to the code generator for improved stack and workspace usage.

Floating point support

Special floating point instructions are generated to exploit the speed of the built in floating point unit of the T8. For the T4 target floating point operations are emulated in software.

Ada Run Time

The Ada Run Time Executive provides complete and efficient support for executing Ada programs. Thorough and predictable storage reclamation is implemented, minimizing the application's chance of raising STORAGE_ERROR. Absolutely no execution overhead is associated with exceptions unless one is raised; call, return and block entry, exit sequences are free of exception management code and consequently very compact. Tasking support includes pragma PRIORITY, preemptive scheduling, user controllable time slicing and fairness in selective wait.

Supports low level programming features

Representation clauses to the bit level, address clauses, pragma PACK, and unchecked conversion and deallocation are among the supported features. An interface to occam 2 is provided, facilitating the development of true distributed multi-processor applications.

6.2.4 Increased development productivity

Detailed error messages are provided with clear diagnostic information and optionally, explanations and suggested fixes. Run-time error diagnostics include a full trace-back with complete calling sequence and source line information.

AdaXref provides comprehensive cross reference documentation for Ada purposes. AdaReformat can be used to automatically reformat Ada source to the style employed in the Ada reference manual, thus imposing consistency and readability of source files. AdaMake supports automatic rebuilding of Ada programs.

6.2.5 Advanced debugging support

AdaProbe is a combined source-level symbolic debugger and program viewer. It is in part a source-level symbolic debugger able to handle all Ada features, including generics and multi-tasking. Supported features include breakpoints, single-stepping and fine control over exceptions. You can also view the state of the program; AdaProbe keeps a trace of the call history and the values of all variables can be displayed and modified. AdaProbe also supports debugging at the assembly level; you can single-step through individual instructions and display and change memory locations and registers.

Alsys products, unlike many others, do not insert debug code in the executable code, which would distort its performance during testing. All information required for debugging is stored by the compiler in the repository. The Alsys approach guarantees reliable and reproducible debugging.

6.2.6 Ada predefined Input and Output

Predefined Ada Input and Output packages are supported, and performed through the INMOS iserver.

6.2.7 Transfers the loadable Ada program to the target

The application can be executed on any T8 host. A single image can be downloaded to a single T8 or T4, or a network of transputers. Downloading is achieved via the host T8 or by transfer to any iserver supported computer for execution on the target configuration.

6.3 The Alsys Ada Compilation System

The Alsys Ada compilation system is shown in figure 6.1.

6.4 Ada Compiler Toolset Product Components

6.4.1 Documentation

Alsys user documentation is comprehensive, professionally written and easy to use.

User's Guide

The User's Guide explains the use of the Compiler and Binder, detailing the procedures for compiling, binding and linking, and executing the object code. Additional User's Guides are provided with the optional Ada Toolset.

Project Development Guide

The Project Development Guide explains the Alsys Multi-Library Environment, including the use of the Family Manager, Library Manager and Unit Manager.

Installation Guide

The Installation Guide specifies the installation procedure.

Ada Reference Manual

The Ada Reference Manual defines the ANSI/ISO standard for the Ada Language (ANSI/MIL-STD 1815A).

Appendix F

Appendix F accompanies the Ada Reference Manual and describes implementation dependent issues and interfaces to other languages.

Application Developer's Guide

The Application Developer's Guide explains how to build an application and use pragma INTERFACE to occur 2.

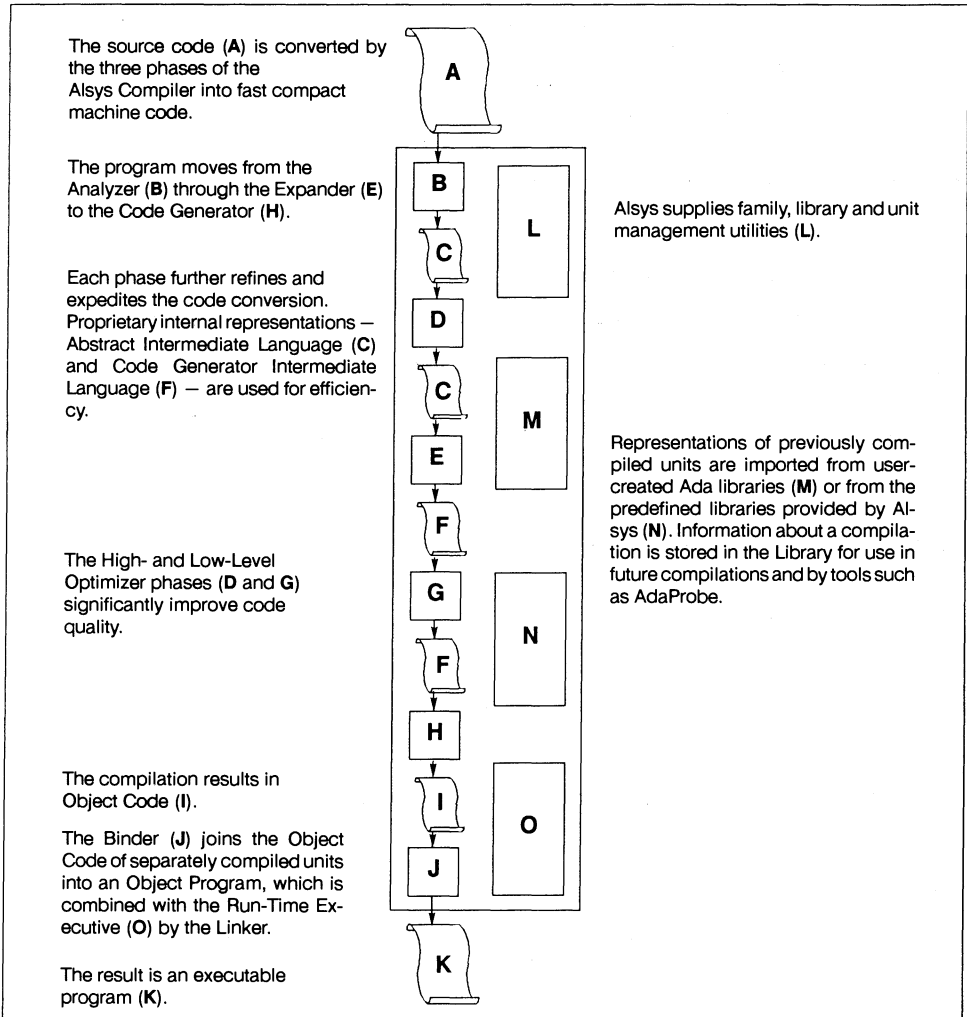


Figure 6.1 The Alsys compilation system

6.4.2 Software Components

- | | |
|-----------------------------|--------------------------|
| 1 Compiler | 2 High Level Optimiser |
| 3 Low Level Optimiser | 4 Binder |
| 5 Multi-Library Environment | 6 Ada Run-Time Executive |
| 7 AdaWorld | 8 AdaProbe |
| 9 AdaXref | 10 AdaMake |
| 11 AdaReformat | |

6.5 Product Variants

6.5.1 IBM PC Alsys Ada Compiler

Although the PC tools are invoked as if they were ordinary PC resident tools, they actually run on the transputer board plugged into the PC.

Operating requirements

- IBM PC XT or AT with 512Kbyte memory
- IMS B008 Motherboard
- Transputer module with 4 Mbytes of memory
- DOS 3.0 or later
- 30 Mbyte of free disk space
- INMOS IBM PC occam 2 toolset

Distribution media

Software is distributed on **BOTH** 1.2 Mbyte (48TPI) 5.25 inch IBM format floppy disks **AND** 1.44 Mbyte 3.5 inch IBM format floppy disks.

6.5.2 VAX VMS Alsys Ada Compiler

All tools are provided in a form which will run on the host machine.

Operating requirements

- Any VAX, MicroVAX or VAXstation
- VAX VMS 4.7 or later
- 30 Mbytes of free disk space
- INMOS VAX VMS occam 2 toolset

Distribution media

Software is distributed on a TK50 tape cartridge.

6.6 Customer Support And Upgrade Services

Alsys offers several levels of Customer Support and Upgrade Services to satisfy varied customer needs. Services include access to the electronic Alsys Customer Support Bulletin Board, automatic shipment of new releases on supported media, multiple levels of telephone support and consulting. Support is available worldwide through offices located in the United States, the United Kingdom and France.

6.7 Alsys And Ada

In 1980, after winning the historic international design competition for the creation of the Ada language itself, Dr. Jean D. Ichbiah founded Alsys. Many members of the original design team are employed at Alsys companies in the United States, United Kingdom and France. Together the companies employ over 150 people, more than 80 of whom are software developers.

Alsys offers a complete range of Ada products. AlsyEd education products include videotaped and live Ada courses and computer aided instruction. AlsyComp compiler products and AslyTool toolsets are available for a broad range of popular architectures including the Inmos transputer processor families.

For ordering information please contact the following:

Alsys Offices and Addresses

Alsys, Inc.
67 South Bedford Street
Burlington, MA 01803-5152
Tel: (617) 270-0030
Fax: (617) 270-6882

Alsys, Inc.
Southeast Regional Office
1800 Alexander Bell Drive
Suite 102
Reston, VA 22091
Tel: (703) 391-0771
Fax: (703) 391-0470

Alsys, Inc.
Western Regional Office
23282 Mill Creek Road
Suite 211
Laguna Hills, CA 92653
Tel: (714) 472-2410
Tel: (714) 472-2414

Alsys, S.A.
29, Avenue de Versailles
78170 La Celle Saint-Cloud
France
Tel: 33 (1) 39 18 12 44
Telex: 697569F
Fax 33 (1) 39 18 26 80

Alsys, Ltd.
Partridge House
Newtown Road
Henley-on-Thames
Oxon RG9 1EN, England
Tel: 44 (491) 579090
Telex: 846508 ALSHEN G
Fax: 44 (491) 571866

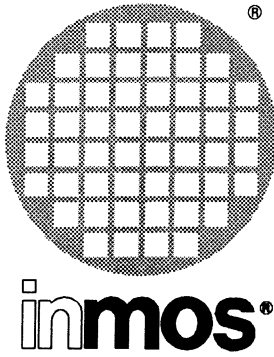
Alsys AB
Patron Pehrs Väg 10
Box 1085
141 22 Huddinge / Stockholm
Sweden
Tel: 46 (8) 746 0920
Fax: 46 (8) 774 5213

Alsys GmbH
Am Rüppurrer Schloss 7
D-7500 Karlsruhe 51
Germany
Tel: 49 (721) 883025
Fax: 49 (721) 887564

ALSYS is a trademark of ALSYS S.A.



Transputer Development Kits



Transputer Development Kits

Product Information

7.1 Transputer Development Kits

The success of any microprocessor is determined as much by the quality of the available development tools as by any other feature. Silicon performance is intimately linked to compiler technology. The use of efficient debugging tools is one of the most effective ways to reduce time to market. This is why INMOS produces a complete range of integrated development tools, specifically designed for multi-processing applications.

All development tools are designed specifically with multiprocessing in mind. For example, configurers automatically add bootstrap code to load a complete network of processors, and the interactive debugger allows the user to 'walk down a link' and inspect/modify the processor state at the other end.

INMOS is able to support transputer application development on a wide range of standard computer platforms including:

- IBM PC-AT or compatible
- NEC 9800 series PC
- IBM PS/2
- SUN3
- SUN386i
- SUN4
- VAX VMS

The tools are designed such that there is a consistent user interface, enabling users to migrate from one environment to another with the minimum of effort.

There follows a series of tables designed to help the reader select the components that are required to build a suitable transputer development environment.

Table 7.1 indicates the part numbers of the different language tools available for each of the supported computer platforms. The C++ tools are pre-processors. All the other language variants (i.e. ANSI C, FORTRAN and OCCAM) are complete toolsets including compiler, configurer, debuggers and a host of other utilities.

	PC	NEC PC	PS/2	Sun3	Sun386i	Sun4	VAX
ANSI C	IMS D7214	IMS D7214	IMS D7214	IMS D5214	IMS D7214	IMS D4214	IMS D6214
C+ +	IMS D7217	IMS D7217	IMS D7217	IMS D5217	IMS D7217	IMS D4217	IMS D6217
Fortran	IMS D7216	IMS D7216	IMS D7216	IMS D5216	IMS D7216	IMS D4216	IMS D6216
occam	IMS D7205	IMS D7205	IMS D7205	IMS D5205	IMS D7205	IMS D4205	IMS D6205

Table 7.1

Two modes of development are possible using INMOS tools. For example, in some cases it is possible to run the compiler on the host computer, generating binary code to run on an attached transputer system (or on a simulator). This is known as 'cross-development'.

Alternatively, in all cases it is possible to run the compiler directly on the attached transputer system. This is known as 'transputer hosted' operation.

Table 7.2 shows which modes of operation are supported on which computer platforms. Table 7.3 indicates the hardware and software components that are required to build a transputer system that can be directly interfaced to the platform, and is capable of running the development tools in 'transputer-hosted' mode.

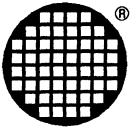
	PC	NEC PC	PS/2	Sun3	Sun386i	Sun4	VAX
Cross-development	No	No	No	Yes	No	Yes	Yes
Transputer hosted	Yes	Yes	Yes	Yes	Yes	Yes	Yes

Table 7.2

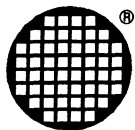
Host Machines						
	PC	NEC PC	PS/2	Sun3	Sun386i	Sun4
Motherboard	IMS B008-1	IMS B015-1	IMS B017-1	IMS B014-1	IMS B008-1	IMS B014-1
Host TRAM¹	IMS B404-3	IMS B404-3	IMS B404-3	IMS B404-3	IMS B404-3	IMS B404-3
Debug TRAM¹	IMS B404-3	IMS B404-3	IMS B404-3	IMS B404-3	IMS B404-3	IMS B404-3
Card frame	—	—	—	IMSCA12	—	IMSCA12
Device driver	Included	Included	IMSS217	IMSS514	IMSS308	IMSS514
Notes						
1 Any TRAM with 2Mbytes or more of memory is sufficient. An IMS B404-3 is suggested but an IMS B428-12 (very fast 2Mbyte), IMS B426-5 (4Mbyte), or an IMS B427-5 (8 Mbyte) could equally well be used.						

Table 7.3

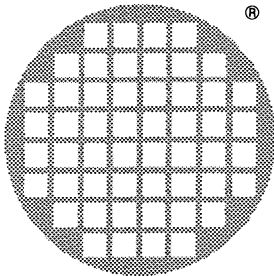
For networked computer systems running TCP/IP, INMOS is also able to offer software and hardware to support remote access to transputer systems. The reader is referred to the sections on the IMS B300 (chapter 44) and on Ethernet support software (chapter 14).



Systems Software



Board Support Software



IMS F000B

VecTRAM library



Product Information

A software support package for the IMS B420 Vector Processing TRAM

KEY FEATURES

- Dramatically speeds up parallel applications and increases system performance involving vector/signal processing computation.
- occam 2 and ANSI C compatible(IMS Dx205 and IMS Dx214 respectively).
- Obviates the need for users to directly program the hardware of IMS B420 for common vector/signal processing operations.
- When a vector library function is encountered, the co-processor is activated to execute the required function transparently
- Automatically copies data structures from non shared memory areas to a shared memory area accessible by both the transputer and the Zoran ZR34325 vector co-processor.

APPLICATIONS

- Speech processing
- Communication and coding
- Graphics and numerical processing
- Radar, sonar, ultrasonics, etc.
- Seismic/geophysical data processing
- Neural networks
- Image processing and compression

8.1 Introduction

The IMS F000B consists of a library of C functions and OCCam procedures developed specifically for common vector/signal processing tasks, encountered in many applications. The functions are callable from a C or an OCCam program running on an IMS B420 TRAM, and can be used to dramatically speed up parallel applications and system performance involving vector/signal processing computation.

This document is a brief introduction to the features provided by IMS F000B and explains, by way of example, how the product may be used.

It is assumed that the reader is familiar with :

- Transputers, OCCam and C
- Digital Signal Processing (DSP)
- IMS Dx205 OCCam 2 toolset [1]
- IMS Dx214 ANSI C toolset [2]

8.2 Product Overview

During program execution, when a vector library function is encountered, the co-processor is activated to execute the required function. On the IMS B420 TRAM, the co-processor has its own local fast memory space, which is also accessible to the transputer. However the transputer local memory is not visible to the co-processor. When a library function is called; if the data to be processed is in the transputer local memory space, it is automatically copied (using the block move capability of the T800) to a predetermined area in the co-processor space. The co-processor is then activated to execute the required function.

If the destination vector operand address, specified in the call, is in the transputer space, the processed data is automatically copied back to the specified area in the transputer memory space. This built-in copying means that the co-processor operation can be totally transparent to the programmer, and programs can be accelerated without the need for detailed knowledge about the operation of the IMS B420 TRAM.

The overhead associated with data copying, between the transputer and co-processor (or visa versa), is avoided, if the source and/or destination operands for the specified function are already in the co-processor local memory space. The library functions automatically check the operand addresses and take appropriate action. In general, if the address of an input or an output operand, in a function call, is in the co-processor space, no data copying will take place for that operand. This is particularly important if operands are to undergo several vector/signal processing operations. For optimal performance, in such cases, the user can easily specify destination (and/or source) addresses which are local to the co-processor address space. In this way data copying, between the transputer and the shared memory area, can be minimised.

Apart from vector and arithmetic functions, the IMS F000B includes efficient vector move functions which allow optimisation at the application level. The library also supports co-processor control calls which are used to set rounding modes, etc.

8.3 Using IMS F000B

IMS F000B may be used by both C and OCCam 2 programs. Throughout this document all descriptions of parameters will use C calling conventions. Hence OCCam users should read ‘_’ as ‘.’.

Before using the supplied library routines, a call **must** be made to the initialisation routine **VT_INIT (vt.init)**, which initialises the Vector Co-processor.

Simple examples in both OCCam and C are given for a call to the vector multiplication routine **VT_MULT_F32R (vt.mult.f32r)**.

The vector multiplication routine multiplies two data vectors pointed to by `VecIn1` and `VecIn2` and places the product vector at the location pointed to by `VecOut`.

The vector operands are 32-bit floating-point throughout. The elements in the input vectors will have strides of `VecIn1_stride`, and `VecIn2_stride` respectively. The elements of the output vector will be stored with strides of `VecOut_stride`.

Operation:

$$\text{VecOut}[i \times \text{VecOut_stride}] = \text{VecIn1}[i \times \text{VecIn1_stride}] \times \text{VecIn2}[i \times \text{VecIn2_stride}]$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

Calling Syntax:

(C-)

Function: VT_MULT_F32R

```
int VT_MULT_F32R (float *VecIn1,
                 int    VecIn1_stride,
                 float *VecIn2,
                 int    VecIn2_stride,
                 float *VecOut,
                 int    VecOut_stride,
                 int    No_of_elements,
                 int    Flag);
```

(Occam:)

PROC: vt.mult.f32r

```
vt.mult.f32r (VAL INT    gsb,
              []REAL32  VecIn,
              VAL INT    VecIn1.stride,
              []REAL32  VecIn2,
              VAL INT    VecIn2.stride,
              []REAL32  VecOut,
              VAL INT    VecOut.stride,
              VAL INT    No.of.elements,
              VAL INT    Flag,
              INT        error.code)
```

Notes:

The `gsb` parameter is required to permit OCCAM to use of the supplied library **VECC14.LIB** which contains compiled C functions. The `gsb` parameter is initialised by `vt.init`.

Errors:

Successful completion returns 0. If an error occurs a non-zero value will be returned.

8.3.1 Incorporation into a C program

A C program, running on VecTRAM, can have the following form:

```
#include"decc14.h"
#include"memc14.h"

main()
{
float a[100], b[100], c[100];
int i, flag, error_code;

/* Initialise the coprocessor
and set up workspaces */
error_code = VT_INIT (COPROCESSOR_MEM_BASE,
LIB_WORKSPACE_BASE,
USER_WORKSPACE_BASE,
COPROCESSOR_MEM_TOP);

if (error_code == 0)
{
/* Initialise test data arrays */
for(i=0; i<100; i++)
{
a[i] = 10.0;
b[i] = 20.0;
c[i] = 0.0;
}

flag =0;

/* Call vector multiply function */
error_code = VT_MULT_F32R(a, 1, b, 1, c, 1, 100, flag);
}
```

8.3.2 Incorporation into an OCCAM program

An OCCAM program, running on VecTRAM, can have the following form:

```
#OPTION"v" -- Disable separate vector space usage
#include"vtc2oc.inc"

PROC ocexample()

#USE "vecc14.lib"
#USE "vtinitoc.t8x"

INT error.code:
[100]INT32 a, b, c :
VAL INT flag IS 0:

--Initialise the C runtime system and the

SEQ
vt.init(gsb, error.code)

--Initialise test data arrays
IF (error.code = 0)
SEQ
```

```

SEQ i = 0 FOR 100
  SEQ
    a[i] := 10.0 (REAL32)
    b[i] := 20.0 (REAL32)
    c[i] := 0.0 (REAL32)

  --Call vector multiply function
  vt.mult.f32r(gsb, a, 1, b, 1, c, 1, 100, flag,
    error.code)

TRUE
SKIP

```

8.4 Supplied Routines

The following list of C functions and OCCAM procedures, are provided by the library **VECC14.LIB**. Each routine will be briefly described.

Operation	C Function	occam procedure
Absolute-Real	VT_ABS_F32R	vt.abs.f32r
Addition-Real	VT_ADD_F32R	vt.add.f32r
Compare-Real	VT_CMP_F32R	vt.cmp.f32r
Disable Co-processor Error Flags	VT_DISERROR	vt.diserror
Division-Real	VT_DIV_F32R	vt.div.f32r
Dot Product-Complex	VT_DOT_F32C	vt.dot.f32c
Dot Product-Real	VT_DOT_F32R	vt.dot.f32r
Enable Co-processor Error Interrupts	VT_ENERROR	vt.enerror
Fast Fourier Transform Complex	VT_FFT_F32C	vt.fft.f32c
FIR Filter	VT_FIR_F32R	vt.fir.f32r
Floating Point to Integer Conversion	VT_F32TOI16_R	vt.f32toi16r
IIR Filter-Complex	VT_IIR_F32C	vt.iir.f32c
IIR Filter-Real	VT_IIR_F32R	vt.iir.f32r
Matrix Multiplication - Complex	VT_GMTX_F32C	vt.gmtx.f32c
Matrix Multiplication - Real	VT_GMTX_F32R	vt.gmtx.f32r
Inverse Fast Fourier Transform-Complex	VT_IFFT_F32C	vt.ifft.f32c
Integer to Floating Point Conversion	VT_I16TOF32_R	vt.i16tof32r
Log10	VT_LOG10_F32R	vt.log10.f32r
Magnitude-Complex	VT_MAG_F32C	vt.mag.f32c
Magnitude Square - Complex	VT_MAGSQ_F32C	vt.magsq.f32c
Max - Real	VT_MAX_F32R	vt.max.f32r
Mean - Real	VT_MEAN_F32R	vt.mean.f32r

Operation	C Function	occam procedure
Min - Real	VT_MIN_F32R	vt.min.f32r
Move Bytes	VT_MOV_BYTE	vt.mov.byte
Move Words	VT_MOV_WORD	vt.mov.word
Multiply - Complex	VT_MULT_F32C	vt.mult.f32c
Multiply - Real	VT_MULT_F32R	vt.mult.f32r
Power - Real	VT_POWER_F32R	vt.power.f32r
Rounding	VT_ROUNDMODE	vt.roundmode
Scale - Real	VT_SCALE_F32R	vt.scale.f32r
Square Root - Real	VT_SQRT_F32R	vt.sqrt.f32r
Vector Subtract	VT_SUB_F32R	vt.sub.f32r

8.4.1 Vector Absolute Value - Real

Operation:

```
VecOut[i x VecOut_stride] = |VecIn[i x VecIn_stride]|
```

8.4.2 Vector Addition - Real

Operation:

```
VecOut[i x VecOut_stride] =  
VecIn[i x VecIn1_stride] + VecIn2[i x VecIn2_stride]
```

FOR i = 0, 1, . . . , No_of_elements

8.4.3 Vector Compare - Real

Operation:

```
FOR i = 0, 1, . . . , No_of_elements
```

```
IF VecIn1[i x VecIn1_stride] . VecIn2[i x VecIn2_stride]|
```

```
THEN VecOut[i x VecOut_stride] = 1.0
```

```
ELSE VecOut[i x VecOut_stride] = 0.0
```

8.4.4 Disable Co-processor Error Flags

Description:

The co-processor operation is normally interrupted on *invalid* operations and also those that have caused an *overflow*. These defaults can be changed using the routine Enable Coprocessor Error Flags as described in section 8.4.8. The VT_DISERROR routine disables the specified error interrupt and allows the co-processor to continue despite such an error. Possible values are:

Error Type	Description
'i'	disable invalid operation error interrupts
'o'	disable overflow error interrupts
'u'	disable underflow error interrupts

8.4.5 Vector Division – Real

Operation:

$$\text{VecOut}[i \times \text{VecOut_stride}] = \frac{\text{VecIn}[i \times \text{VecIn1_stride}]}{\text{PVecIn2}[i \times \text{VecIn2_stride}]}$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.6 Vector Dot Product – Complex

Operation:

$$\begin{aligned} \text{Re}\{\text{DotProd}\} = & \sum_{i=0}^{n-1} \text{Re}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ & \times \text{Re}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \\ & - \\ & \text{Im}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ & \times \text{Im}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \end{aligned}$$

$$\begin{aligned} \text{Im}\{\text{DotProd}\} = & \sum_{i=0}^{n-1} \text{Re}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ & \times \text{Im}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \\ & + \\ & \text{Im}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ & \times \text{Re}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \end{aligned}$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.7 Vector Dot Product – Real

Operation:

$$\text{DotProd} = \sum_{i=0}^{n-1} \text{VecIn1}[i \times \text{VecIn1_stride}] \times \text{VecIn2}[i \times \text{VecIn2_stride}]$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.8 Enable Co-processor Error Interrupts

Description:

The co-processor operation is normally interrupted on *invalid* operations and also those that have caused an *overflow*.

This routine allows these defaults to be modified. It enables co-processor interruption on errors of type `Error` type. If such errors occur during a vector library call, the value returned by the vector function specifies the type of error. Possible values are:

Error Type	Description
'i'	Enable invalid operation error interrupts
'o'	Enable overflow error interrupts
'u'	Enable underflow error interrupts

8.4.9 Fast Fourier Transform – Complex

Description:

The Fast Fourier Transform routine implements a Radix-2 decimation in time FFT on a complex data vector pointed to by `VecIn`. The real and imaginary data elements are stored in successive locations, starting with first real component. This means that a complex data value takes up two successive memory locations, with the lower address holding the real part and the higher address holding the imaginary part. Users must observe this convention, it is good practice to use even and odd addresses for real and imaginary components respectively.

The FFT routine, supplied in this release, can implement FFTs on vector lengths which are integer powers of two, up to 1024. That is possible vector lengths are 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 (complex data). Standard decomposition techniques can be used to convert larger FFTs into a set of smaller ones within the above range.

Operation:

$$\text{VecOut}_{\text{Complex}}[i \times \text{VecOut_stride}] = \sum_{i=0}^{n-1} \text{VecIn}_{\text{Complex}}[i \times \text{VecIn_stride}] \times \exp\left(-jki \frac{2\pi}{N}\right)$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

and

$k = 0, 1, \dots, (\text{No_of_elements} - 1); i = 0, 1, \dots, (\text{No_of_elements} - 1)$

8.4.10 Finite Impulse Response (FIR) Filter

Description:

This function applies an FIR filter of length `Filter_size` with an optional decimation factor, `Decim`, to the input data pointed to by `VecIn`. The input vector can have a stride of `VecIn_stride`. The filter coefficients are stored in an area of memory pointed to by `Coeff`. The total number of processed (filtered) output samples are specified by `No_of_output_samples`.

`No_of_output_samples` must be in the range 2..64.

Operation:

$$\text{DataOut}[j] = \sum_{i=0}^{\text{Filter_size}} \{\text{VecIn}[(i + \text{Decim} \times j) \times \text{VecIn_stride}] \times \text{coeff}[i]\}$$

For

$j = 0, 1, 2, \dots, (\text{No_of_output_samples} - 1)$

8.4.11 Vector Floating Point to Integer (16-bit) Conversion.

Operation:

`VecOut[i × VecOut_stride] = int(VecIn[i × VecIn_stride])`

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.12 Infinite Impulse Response (IIR) Filter – Complex.

Description:

This function applies an IIR filter of length `Filter_size` to the complex input data pointed to by `VecIn`. The input vector can have a stride of `VecIn_stride`. The filter coefficients are stored in an area of memory pointed to by `Coeff`. The total number of processed (filtered) output samples are specified by `no_of_output_samples`.

Operation:

Complex filter:
$$\text{VecOut}(z) = \text{VecIn}(z) \cdot \prod_{k=1}^{n/2} \frac{z + b_k}{z + a_k}$$

Note: Coefficients are loaded as a vector as follows:

`coeffa1(1), coeffb1(1), coeffa1(2), coeffb1(2), ...`

8.4.13 Infinite Impulse Response (IIR) Filter – Real.

Description:

This function applies an IIR filter of length `Filter_size` to the input data pointed to by `VecIn`. The input vector can have a stride of `VecIn_stride`. The filter coefficients are stored in an area of memory pointed to by `Coeff`. The total number of processed (filtered) output samples are specified by `no_of_output_samples`.

Operation:

Real filter:
$$\text{VecOut}(z) = \text{VecIn}(z) \cdot \prod_{k=1}^{n/2} \frac{z^2 + b1_k \cdot z + b2_k}{z^2 + a1_k \cdot z + a2_k}$$

Note: Coefficients are loaded as follows:

`coeffb2(1), coeffa2(1), coeffb1(1), coeffa1(1), coeffb2(2), coeffa2(2), coeffb1(2), coeffa1(2), ...`

8.4.14 Matrix Multiplication – Complex

Operation:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{k1} & b_{k2} & \dots & b_{kn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \dots & \dots & \dots & \dots \\ c_{k1} & c_{k2} & \dots & c_{kn} \end{bmatrix}$$

$$c_{ij} = a_{i1}.b_{1j} + a_{i2}.b_{2j} + \dots + a_{in}.b_{nj}$$

where a,b and c are complex values

Figure 8.1 Generic Matrix Multiplication-Complex

8.4.15 Matrix Multiplication – Real

Operation:

$$\begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{k1} & a_{k2} & \dots & a_{kn} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{k1} & b_{k2} & \dots & b_{kn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \dots & c_{1m} \\ c_{21} & c_{22} & \dots & c_{2m} \\ \dots & \dots & \dots & \dots \\ c_{k1} & c_{k2} & \dots & c_{kn} \end{bmatrix}$$

$$c_{ij} = a_{i1}.b_{1j} + a_{i2}.b_{2j} + \dots + a_{in}.b_{nj}$$

Figure 8.2 Generic Matrix Multiplication-Real

8.4.16 Inverse Fast Fourier Transform – Complex

Description:

The Inverse Fast Fourier Transform routine implements a Radix-2 decimation in time inverse FFT on a complex data vector pointed to by `VecIn`. The real and imaginary data elements are stored in successive locations, starting with first real component. This means that a complex data value takes up two successive memory locations, with the lower address holding the real part and the higher address holding the imaginary part. Users must observe this convention, it is good practice to use even and odd addresses for real and imaginary components respectively.

The IFFT routine, supplied in this release, can implement IFFTs on vector lengths which are integer powers of two up to 1024. That is possible vector lengths are 2, 4, 8, 16, 32, 64, 128, 256, 512 and 1024 (complex data).

Operation:

$$\text{VecOut}_{\text{Complex}}[i \times \text{VecOut_stride}] = \sum_{k=0}^{n-1} \text{VecIn}_{\text{Complex}}[i \times \text{VecIn_stride}] \times \exp\left(-jk \frac{2\pi}{N}\right)$$

WHERE $n = \text{No_of_elements}$

and

$k = 0, 1, \dots, (\text{No_of_elements} - 1); i = 0, 1, \dots, (\text{No_of_elements} - 1)$

8.4.17 Vector Integer(16-bit) to 32-bit floating-point Conversion

Operation:

$\text{VecOut}[i \times \text{VecOut_stride}] = \text{float}(\text{VecIn}[i \times \text{VecIn_stride}])$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.18 Vector Log to the base 10 - Real**Operation:**

$$\text{VecOut}[i \times \text{VecOut_stride}] = \log_{10}(\text{VecIn}[i \times \text{VecIn_stride}])$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.19 Vector Magnitude - Complex**Operation:**

$$\text{VecOut}[i \times \text{VecOut_stride}] \approx \sqrt{(\text{Re}\{\text{VecIn}[i \times \text{VecIn_stride}]\})^2 + (\text{Im}\{\text{VecIn}[i \times \text{VecIn_stride}]\})^2}$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.20 Vector Magnitude Square - Complex**Operation:**

$$\text{VecOut}[i \times \text{VecOut_stride}] = (\text{Re}\{\text{VecIn}[i \times \text{VecIn_stride}]\})^2 + (\text{Im}\{\text{VecIn}[i \times \text{VecIn_stride}]\})^2$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.21 Find the Element with the Maximum Value and Its Position - Real**Operation:**

$$\text{MaxOut} = \text{MAXIMUM}(\text{VecIn}[i \times \text{VecIn_stride}])$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

and

CountOut = (j) WHERE $j = i$ for the element with the maximum value

8.4.22 Vector Mean - Real**Operation:**

$$\text{MeanOut} = (1/N) \sum_{i=0}^{n-1} \text{VecIn}[i \times \text{VecIn_stride}]$$

WHERE $N = \text{No_of_elements}$

and

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.23 Find the Element with the Minimum Value and Its Position - Real**Operation:**

$$\text{MinOut} = \text{MINIMUM}(\text{VecIn}[i \times \text{VecIn_stride}])$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

and

CountOut = (j) WHERE j = i for the element with the minimum value

8.4.24 Vector Move - Bytes

Description:

This routine block moves bytes in memory.

8.4.25 Vector Move - Words (32-bit)

Description:

This routine block moves words in memory.

8.4.26 Vector Multiply - Complex

Operation:

$$\begin{aligned} \text{Re}\{\text{VecOut}[i \times \text{VecOut_stride}]\} &= \text{Re}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ &\quad \times \text{Re}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \\ &\quad - \\ &\quad \text{Im}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ &\quad \times \text{Im}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \\ \text{Im}\{\text{VecOut}[i \times \text{VecOut_stride}]\} &= \text{Re}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ &\quad \times \text{Im}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \\ &\quad - \\ &\quad \text{Im}\{\text{VecIn1}[i \times \text{VecIn1_stride}]\} \\ &\quad \times \text{Re}\{\text{VecIn2}[i \times \text{VecIn2_stride}]\} \end{aligned}$$

FOR i = 0, 1, ..., No_of_elements

8.4.27 Vector Multiply - Real

Operation:

$$\text{VecOut}[i \times \text{VecOut_stride}] = \text{VecIn1}[i \times \text{VecIn1_stride}] \times \text{VecIn2}[i \times \text{VecIn2_stride}]$$

FOR i = 0, 1, ..., No_of_elements

8.4.28 Vector Power - Real

Operation:

$$\text{VecOut}[i \times \text{VecOut_stride}] = \{\text{VecIn}_1[i \times \text{VecIn_stride}_1]\}^{\text{Exponent}}$$

FOR i = 0, 1, ..., No_of_elements

8.4.29 Modify Co-processor Rounding Mode

Description:

This routine can be used to modify the default rounding mode of the co-processor which is set to rounding to even. The possible values are:

Option	Description
'e'	round to even
'z'	round to zero
'+'	round to $+\infty$
'-'	round to $-\infty$

8.4.30 Vector Scale - Real

Operation:

$$\text{VecOut}[i \times \text{VecOut_stride}] = \text{VecIn}_1[i \times \text{VecIn_stride}] \times (\text{Scale_factor})$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.31 Vector Square Root - Real

Operation:

$$\text{VecOut}[i \times \text{VecOut_stride}] = \sqrt{(\text{VecIn}[i \times \text{VecIn1_stride}])}$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.4.32 Vector Subtract - Real

Operation:

$$\text{VecOut}[i \times \text{VecOut_stride}] = \text{VecIn}_1[i \times \text{VecIn1_stride}] - \text{VecIn}_2[i \times \text{VecIn2_stride}]$$

FOR $i = 0, 1, \dots, \text{No_of_elements}$

8.5 Environment

IMS F000B software is supplied in binary form and is compatible with IMS Dx205 OCCaM 2 and IMS Dx214 ANSI C toolset products.

For program execution, the user will require:

- An IMS B420 VecTRAM
- A suitable TRAM motherboard on which to mount the above

8.6 IMS F000B Product Components

8.6.1 Distribution media

The IMS F000B software is distributed on two media systems:

- 360Kbyte 5.25 inch IBM format diskettes
- 720Kbyte 3.5 inch IBM format diskettes

8.6.2 Documentation

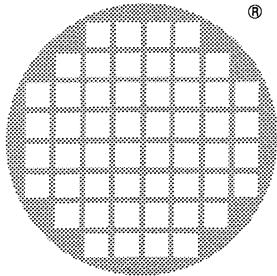
- Delivery manual
- User manual

8.7 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates. Software problem report forms are included with the software. INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.

8.8 References

- 1 OCCaM Toolset datasheet, INMOS Limited, October 1990 (INMOS document number 42 1483 00)
- 2 ANSI C Toolset datasheet, INMOS Limited, August 1990 (INMOS document number 42 1471 00)



inmos[®]

IMS F001B

GPIB libraries

Product Information

A software support package for the IMS B421 GPIB TRAM

KEY FEATURES

- Interfaces transputer networks to GPIB instrumentation
- Performs all commonly required GPIB activities
- Supports GPIB system controller and talker/listener modes
- Maximum sustained transmit rate of 300 Kbytes/second, Maximum sustained receive rate of 90 Kbytes/second
- Compatible with IMS Dx05B and IMS Dx205 OCCam and IMS Dx214 ANSI C.

APPLICATIONS

- Control of scientific test and measurement instrumentation
- Control and processing for Automatic Test Rigs
- Compute-intensive GPIB data processing on an adjacent transputer network
- Embedded systems
- Transputer based GPIB instruments
- Process control
- Data logging

9.1 Introduction

IMS F001B provides a set of routines which allow an application to communicate with GPIB devices without concerning themselves with the low level details of GPIB operation.

IMS F001B must be used in conjunction with the IMS B421 GPIB TRAM.

This document is a brief introduction to the features provided by IMS F001B and explains, by way of example, how the product may be used.

It is assumed that the reader is familiar with:

- Transputers, OCCam and C
- The General Purpose Interface Bus
- The Dx05B OCCam 2 toolset
- The Dx205 OCCam 2 toolset
- The Dx214 ANSI C toolset

9.2 Product Overview

IMS F001B may be used with both OCCam and INMOS ANSI C. There are 2 different library sets provided, one of which is IMS Dx05B OCCam compatible, the other of which is IMS Dx205 OCCam and IMS Dx214 C compatible.

There are 3 distinct sections to the product:

- A process called F001
- A library of GPIB command procedures called F001IO
- 2 channels, via which F001IO procedures communicate with F001

The F001 process performs the following activities:

- Initialising the IMS B421 hardware
- Performing all GPIB activity
- Reading and writing the on-board EEROM
- Accessing the on-board subsystem port

The F001 process must always be resident on the IMS B421 TRAM.

The F001IO library provides the procedural interface to the IMS F001B facilities. The procedures provided in F001IO transfer data and commands to and from the F001 process over the channels mentioned above. All details of the channel communication are hidden from the application.

There are 4 groups of commands provided in the F001IO, which are:

Initialisation

These set user definable features of operation, such as GPIB address, device mode and message termination characteristics.

Device

These can only be used when IMS B421 TRAM is functioning as a GPIB device, and allow it to respond to commands from the bus controller.

Controller

These can only be used when the IMS B421 TRAM is functioning as a GPIB system controller, and allow it to control other devices on the bus.

General

These allow the application to access the on-board EEROM, the subsystem port and the power-on reset latch.

Due to the channel communication described above, the GPIB application may be split across two TRAMs, one of which must be the IMS B421 TRAM. The other TRAM may provide extra processing resource or memory, if that provided by the IMS B421 is insufficient. In this split configuration, the channels are implemented on one link of the IMS B421. Alternatively, the IMS B421 TRAM can provide both the GPIB interface and the complete processing resource. In this case, the channels are implemented in on-board RAM. Code written to run solely on the IMS B421 may be altered to run on two TRAMs merely by altering the configuration file describing the system. Examples showing both types of configurations are given later.

Please note that due to memory requirements, IMS F001B cannot be configured to run solely on the IMS B421 TRAM when used with C. It is necessary to run the F001B libraries on an adjacent TRAM.

9.2.1 IMS F001B command format

This section describes the format and calling syntax for IMS F001B commands. Examples for both OCCam and C are given. The calling syntax for IMS Dx05B and IMS Dx205 OCCam are identical.

All IMS F001B commands take a minimum of three parameters, which are common to all commands. The first two describe the F001-F001IO communication channels. The last is an error parameter, which indicates the termination status of the command. Any other parameters are command specific.

For example, the calling syntax of the RECEIVE command is as follows:

OCCam:

```
PROC F001.RECEIVE (
  CHAN OF F001PROT from.F001, to.F001,
  VAL INT16   talk.address,
  INT32      count,
  []BYTE     received.data,
  INT16      F001.error)
```

C:

```
void f001_receive
(Channel *from_f001, Channel *to_f001,
 short talk_address,
 long *count,
 char received_data[], short rd_size,
 short *f001_error);
```

This command takes five parameters, of which the first two and the last are common to all commands. The other three are specific to the RECEIVE command and respectively describe the talk address of the device which sends the data, the number of bytes received, and the actual data bytes received during the transfer.

The user manual provided with IMS F001B describes all commands in greater detail than is possible here.

Summary of the IMS F001B commands

Initialisation

- END SETUP - complete initialisation phase
- SET BUS DRIVERS - set GPIB bus drivers to tristate / open collector
- SET DEVICE MODE - set mode to system controller or talker/listener
- SET GPIB ADDRESS - set address of IMS B421 TRAM
- SET RECEIVE TERMINATOR - set message terminating condition
- SET TIMEOUT - set byte transmission timeout
- SET TRANSMIT TERMINATOR - set message terminating condition
- START SETUP - begin initialisation phase

Device

- COUNTED RECEIVE RESPONSE MESSAGE - receive N bytes from bus
- GENERATE SERVICE REQUEST - set SRQ true
- RECEIVE BYTE - receive 1 byte from bus
- RECEIVE RESPONSE MESSAGE - receive a data string from bus
- SEND DATA BYTES - send a data string to bus
- WAIT ON GPIB EVENT - monitor bus for various conditions

Controller

- COUNTED RECEIVE - receive N bytes from a named sender
- COUNTED RECEIVE RESPONSE MESSAGE - as above
- DEVICE-DEVICE TRANSFER - transfer data between 2 named devices
- DEVICE CLEAR - IEEE-488.2 Device Clear command
- ENABLE LOCAL CONTROLS - IEEE-488.2 Enable Local Controls command
- ENABLE REMOTE - IEEE.488.2 Enable Remote command
- READ SRQ - read status of SRQ line
- RECEIVE - receive a data string from a named sender
- RECEIVE BYTE - as above
- RECEIVE RESPONSE MESSAGE - as above
- RECEIVE SETUP - address a named device to send
- RESET - IEEE.488.2 Reset command
- SEND - send a data string to a named receiver
- SEND COMMAND - send an ATN true command or address

- SEND DATA BYTES - as above
- SEND IFC - set IFC true
- SEND LLO - IEEE.488.2 Send LLO command
- SEND SETUP - address a named device to SEND
- SERIAL POLL - perform a Serial Poll
- SET REN - IEEE.488.2 Set REN command
- SET RWLS - IEEE.488.2 Set RWLS command
- TRIGGER - IEEE.488.2 Trigger command
- UNTIMED SEND - High speed SEND
- UNTIMED SEND DATA BYTES - High speed SEND DATA BYTES

General

- READ EEROM - read contents of on-board EEROM
- READ POR LATCH - read status of Power On Reset latch
- READ SS ERROR - read SubsystemError signal
- READ 9914 REGISTER - read the registers of the TMS9914A GPIB controller chip
- SET SS ANALYSE - set Subsystem Analyse signal
- SET SS RESET - set SubsystemReset signal
- WRITE EEROM - write to on-board EEROM
- WRITE 9914 REGISTER - write to the registers of the TMS9914A GPIB controller chip

9.2.2 Using IMS F001B

This section explains the usage of IMS F001B by way of example. The source is listed in both OCCAM and C. The aim of the example is two-fold. Firstly, it shows how the IMS F001B libraries are accessed from OCCAM and C, and secondly, it shows how an application may be configured to run on either one or two TRAMs.

The example uses IMS F001B routines to initialise the IMS B421 TRAM, and to send a data string to a named device on the bus.

The following OCCAM code implements the example. Note that the F001IO libraries are accessed by the inclusion of the F001IO.LIB file in the source:

```
#INCLUDE "hostio.inc"
#include "f001.inc"

PROC twotrams (CHAN OF SP fs, ts,
              CHAN OF F001PROT from.F001, to.F001)

#USE "hostio.lib"
#USE "f001io.lib"

-----
PROC DEMO (CHAN OF F001PROT from.F001, to.F001)

--
-- This is a very simple demonstration of some of the F001
-- routines.
--
VAL message IS "hello world from the B421 TRAM ":
BYTE term.byte:
INT16 source, pri.address, mode, drivers, error:
INT16 tx.period, term:
INT32 count:
[1]INT16 la.list:

SEQ
--
-- First, initialize the B421 TRAM - this is mandatory; no commands
-- can be invoked until all the user definable TRAM features have
-- been selected.
--
F001.START.SETUP (from.F001, to.F001, error)
so.write.string (fs, ts, "start.setup error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

pri.address := 1 (INT16)
source := parameter
F001.SET.GPIB.ADDRESS (from.F001, to.F001, source, pri.address, error)
so.write.string (fs, ts, "set.gpib.address error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

source := parameter
mode := system.controller
F001.SET.DEVICE.MODE (from.F001, to.F001, source, mode, error)
so.write.string (fs, ts, "set.device.mode error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

source := default
drivers := tristate
F001.SET.BUS.DRIVERS (from.F001, to.F001, source, drivers, error)
so.write.string (fs, ts, "set.bus.drivers error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)
```

```

tx.period := 1000 (INT16) -- ms
FO01.SET.TIMEOUT (from.FO01, to.FO01, tx.period, error)
so.write.string (fs, ts, "set timeout error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

term := LF.term
FO01.SET.TX.TERMINATOR (from.FO01, to.FO01, term, error)
so.write.string (fs, ts, "set tx terminator error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

FO01.SET.RX.TERMINATOR (from.FO01, to.FO01, term, term.byte, error)
so.write.string (fs, ts, "set rx terminator error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

FO01.END.SETUP (from.FO01, to.FO01, error)
so.write.string (fs, ts, "end setup error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

--
-- pulse IFC true for 128 microseconds - this causes all
-- devices on the bus to terminate any transaction in progress
--
FO01.SEND.IFC (from.FO01, to.FO01, error)
so.write.string (fs, ts, "send ifc error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

--
-- set the REN line true, so all devices enter a remote state
--
FO01.SET.REN (from.FO01, to.FO01, FALSE, error)
so.write.string (fs, ts, "set ren error ")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

--
-- now send an ascii data string to the device at listen address 1
-- (if you wish to change this to another address, alter the constant
-- on the right hand side of the next line of code
--
la.list [0] := 1 (INT16)
count := INT32 (SIZE (message))
FO01.SEND (from.FO01, to.FO01, la.list, count, message, error)
so.write.string (fs, ts, "send error")
so.write.int (fs, ts, INT (error), 10 (INT))
so.write.nl (fs, ts)

so.exit (fs, ts, sps.success)
:

```

SEQ

DEMO (from.FO01, to.FO01)

The following code implements the F001 process, which runs on the IMS B421 TRAM. As described above, this process handles all GPIB activity. The DEMO routine, listed above, communicates with the F001 process over the to.F001 and from.F001 channels. The code for the F001 process itself is accessed via the F001.LIB file.

```
#INCLUDE "f001.inc"

PROC gpibdriver (CHAN OF F001PROT from.F001, to.F001)

    #USE "f001.lib"

    SEQ
        F001 (from.F001, to.F001)
    :
```

These processes are configured to run on two transputers using the standard OCCAM configuration facilities. The DEMO process may be compiled for a T2, T4, or T8 based TRAM, whereas the F001 process may only be compiled for a T2 based TRAM i.e. the IMS B421. The OCCAM toolset documentation should be consulted for further details regarding code configuration.

IMS F001B can also be used from C. The following code fragment shows how IMS F001B routines are accessed when using C. The f001io.h header file provides the function prototypes for the F001IO procedures.

```
#include <misc.h>
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <channel.h>

/* The next header file, f001io.h, provides the prototypes for the F001IO
   routines */

#include <f001io.h>

/* f001def.h provides various constant definitions for error codes
   and general use. You should include it if you wish to use the
   predefined constants provided therein. The f001def.h file should
   be examined for further information */

#include <f001def.h>

void demo_proc (Process *p, Channel *from_f001, Channel *to_f001)
{
    #define ADDRESS_SIZE 1
    #define BUFFER_SIZE 1

    short error, source, gpib_addr, device_mode, driver_type, tx_period,
           term, i, count;

    char term_char;
    char buffer [BUFFER_SIZE];

    short listeners [NO_OF_PRIMARY_ADDRESSES];
    short address [ADDRESS_SIZE];
```



```
/*
The next statement prevents the compiler generating spurious error
messages about unused variables
*/
p = p;

/*
first initialise the B421 TRAM - this is mandatory; no commands can be
invoked until all the user definable TRAM features have been selected.
*/
f001_start_setup (from_f001, to_f001, &error);
printf ("f001_start_setup error : %d\n", error);

source = PARAMETER;
gpib_addr = 3;
f001_set_gpib_address (from_f001, to_f001, source, gpib_addr, &error);
printf ("f001_set_gpib_address error : %d\n", error);

device_mode = SYSTEM_CONTROLLER;
f001_set_device_mode (from_f001, to_f001, source, device_mode, &error);
printf ("f001_set_device_mode error : %d\n", error);

driver_type = TRISTATE;
f001_set_bus_drivers (from_f001, to_f001, source, driver_type, &error);
printf ("f001_set_bus_drivers error : %d\n", error);

tx_period = 100; /* ms */
f001_set_timeout (from_f001, to_f001, tx_period, &error);
printf ("f001_set_timeout error : %d\n", error);

term = EOI_TERM;
f001_set_tx_terminator (from_f001, to_f001, term, &error);
printf ("f001_set_tx_terminator error : %d\n", error);

term = LF_OR_EOI_TERM;
f001_set_rx_terminator (from_f001, to_f001, term, term_char, &error);
printf ("f001_set_rx_terminator error : %d\n", error);

f001_end_setup (from_f001, to_f001, &error);
printf ("f001_end_setup error : %d\n", error);

... application code goes here
}
```

An equivalent C program to the OCCaM gpib driver code is used to start F001B as a parallel process running on the IMS B421 TRAM. Again, standard configuration utilities provided in the toolset are used to place the application code and the F001B process onto the correct TRAMs.

As mentioned above, when used with INMOS ANSI C, IMS F001B must always be configured to run on two TRAMs, due to memory requirements.

9.3 Operating environment

For program execution, the user will require:

- An IMS B421 TRAM
- A general purpose compute TRAM, such as the IMS B404
- A suitable TRAM motherboard on which to mount the above
- A suitable GPIB - IMS B421 cable

9.4 IMS F001B Product Components

9.4.1 Distribution media

The IMS F001B software is distributed on two media systems:

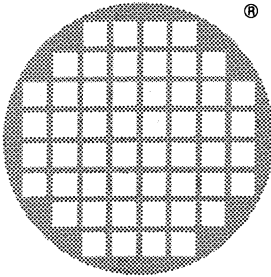
- 360Kbyte 5.25 inch IBM format diskettes
- 720Kbyte 3.5 inch IBM format diskettes

9.4.2 Documentation

- Delivery manual
- User manual

9.5 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates. Software problem report forms are included with the software. IN-MOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.



IMS F002B

SCSI libraries



Product Information

A software support package for the IMS B422 SCSI TRAM

KEY FEATURES

- Connection of SCSI (Small Computer System Interface) peripheral devices to transputer applications.
- occam 2 and ANSI C toolset compatible (IMS Dx205 and IMS Dx214).
- Initiator and Target mode operation.
- Achieves a sustained 1.5 Mbytes/second data transfer rate between a transputer application and a SCSI device (1 transputer link bandwidth).
- Common Command Set supported (SCSI commands for Direct Access Devices).
- Programmable software timeouts for error recovery.
- Programmable SCSI bus data transfer rate.
- Generic interface that permits non supported SCSI commands to be issued to SCSI target devices.
- Diagnostic test upon IMS B422 hardware.
- Example programs of both initiator and target mode operation in both C and occam provided.
- Provides 4 transputer channels to the SCSI Bus (Up to 4 initiator mode applications may share a single IMS B422 SCSI TRAM).

APPLICATIONS

- Embedded systems.
- Creating transputer based SCSI peripheral devices, e.g. Laser printers, scanners, communications devices, vendor unique devices, application accelerators, etc).
- File Systems.
- Disk Arrays that are fault tolerant and/or achieve high performance.
- Optical storage systems (including CD-ROM).
- Computer animation from disk.
- Process control equipment (statistics / data logging)
- Interfacing transputer networks to host computer systems.

10.1 Introduction

The IMS F002B SCSI support software package provides a low level interface between a user application and the IMS B422. The interface presented to the application program is intended to abstract hardware implementation details from the caller, thus minimising the impact of any future hardware upgrades.

This document is intended to be read in conjunction with the IMS B422 SCSI TRAM datsheet [1].

It is assumed that the reader is familiar with :

- Transputers, OCCAM and C
- The SCSI specification [4] and [5]
- IMS Dx205 occam 2 toolset [2]
- IMS Dx214 ANSI C toolset [3]

10.1.1 SCSI overview

The Small Computer System Interface (SCSI), is a local I/O bus that can be operated at data rates up to 4 megabytes per second depending upon circuit implementation choices. The primary objective of the interface is to provide host computers with device independence within a class of devices. Thus different disk drives, tape drives, printers, and communication devices can be added to the host computer(s) without requiring modifications to generic system hardware or software (figure 10.1) . Provision is made for the addition of nongeneric features and functions through vendor unique fields and codes.

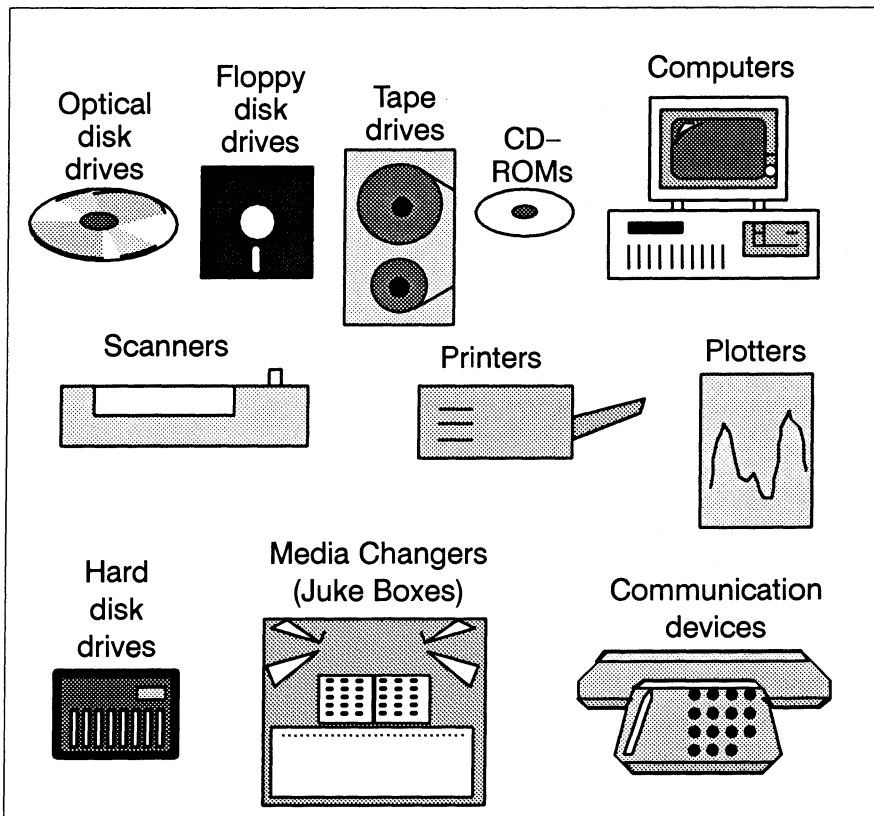


Figure 10.1 Devices available with SCSI interfaces.

10.2 Product Overview

IMS F002B consists of 4 major components :-

- IMS B422 device driver.
- Initialisation interface.
- Initiator mode interfaces (Host computer interface).
- Target mode interfaces (Peripheral operation interface).

The IMS B422 device driver runs on the IMS B422 SCSI TRAM. The other three components will usually be configured upon the user application transputer (they are for the most part in a descheduled condition and therefore consume little processor resources).

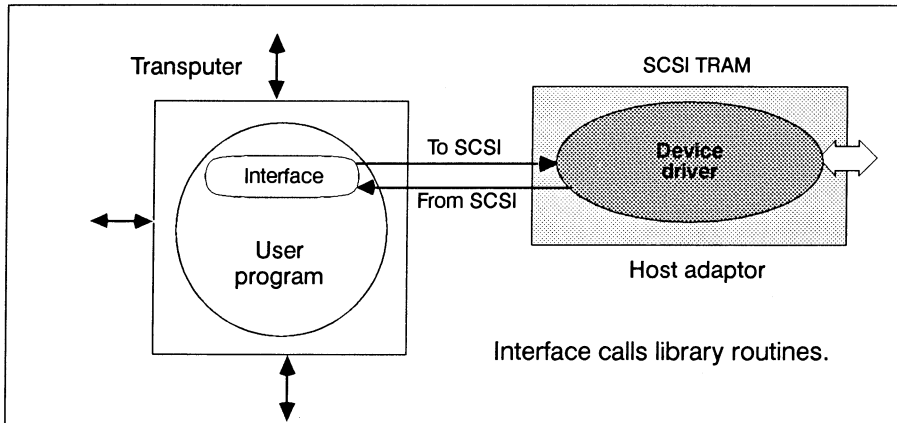


Figure 10.2 Software Elements

10.2.1 IMS B422 Device Driver.

The IMS B422 device driver resides on IMS B422 and is responsible for :-

- Providing the facilities of a Host Adaptor Device Driver for initiator mode interfaces.
- Receiving SCSI requests presented by the SCSI bus and forwarding on the received requests to transputers connected to the 4 transputer links of IMS B422 when in target mode.
- Performing tests upon the hardware of the SCSI interface circuitry and data buffer areas used for SCSI bus data transfer on the IMS B422.

For the purpose of communicating to connected transputers, the IMS B422 device driver uses the channel protocols TO.SCSI (input channel) and FROM.SCSI (output channel). The use of these protocols permits SCSI configuration information, SCSI commands and associated data packets to be transferred across the same transputer link. Optionally, data throughput may be increased by specifying any one of the other unused transputer links of IMS B422 to be an additional data only link to be used in tandem with the currently accessed link. It is not necessary for the application to directly interface at link protocol level.

10.2.2 Initialisation Interface.

The Initialisation interface permits the user to define operating characteristics of the IMS B422. It also permits self test to be performed.

10.2.3 Initiator Mode Interface.

The Initiator mode interfaces provide transputer applications executing upon transputers connected to IMS B422 the ability to access SCSI peripherals via IMS B422.

A generic driver (Host Adaptor Device Driver Interface HADDIF), is provided that accepts SCSI Command sequences and appropriate data buffer areas. The supplied SCSI Command sequence is issued to the IMS B422 device driver for execution by the specified SCSI target device. HADDIF then provides the target device via IMS B422, the ability to access the supplied data areas on the connected transputer in order to provide the initiation requested service. Whilst HADDIF is executing, the CPU resources for SCSI I/O of HADDIF's transputer are maintained at an absolute minimum, thereby giving maximum CPU resources to the user's application.

(occam):

```
SCSI.HADD.IF( CHAN OF TO.SCSI    TO.SCSI.HA,
              CHAN OF FROM.SCSI  FROM.SCSI.HA
              VAL BYTE           Target.ID,
              VAL BYTE           LUN,
              VAL BYTE           SCSI.Command.Length,
              VAL[] BYTE         SCSI.Command,
              VAL BYTE           Direction,
              VAL INT32          Rx.Transfer.Length,
              [] BYTE            Rx.Data,
              VAL INT32          Tx.Transfer.Length,
              VAL[] BYTE         Tx.Data,
              BYTE               Msg.Length,
              [] BYTE            Message,
              BYTE               SCSI.Status,
              INT16              Execution.Status)
```

(C):

```
int scsi_hadd_if( channel      *to_scsi_ha,
                 channel      *from_scsi_ha,
                 char          target_id,
                 char          lun,
                 char          scsi_command_length,
                 char          scsi_command[],
                 char          direction,
                 int           rx_transfer_length,
                 char          rx_data[],
                 int           tx_transfer_length,
                 char          tx_data[],
                 char          *msg_length,
                 char          message[],
                 char          *scsi_status)
```

Common Command Set

Specific support is provided for SCSI commands that are members of the Common Command Set. These SCSI commands can be found on the majority of Direct Access SCSI Winchester disk drives. For each member of the Common Command Set, two interfaces are provided.

The primary interface form accepts non byte packed parameters, transparently builds a byte packed SCSI command sequence and issues it to HADDIF for execution by a specified target device.

A secondary interface form also accepts non byte packed parameters and returns a built byte packed SCSI command sequence. No access is made to IMS B422 by this form of interface. The user must specifically make a call to HADDIF in order to execute the built SCSI command sequence.

Example of primary interface form :-

(occam):

```
PROC SCSI.Read.10( CHAN OF TO.SCSI      TO.SCSI.HA,
                  CHAN OF FROM.SCSI    FROM.SCSI.HA,
                  VAL BYTE              Target.Id,
                  VAL BYTE              LUN,
                  VAL BYTE              dps
                  VAL BYTE              tpa
                  VAL BYTE              reladr
                  VAL INT32             Logical.Block.Address,
                  VAL INT32             Number.of.Blocks,
                  VAL INT32             Block.Size,
                  VAL BYTE              Control.Byte,
                  []BYTE                 Rx.Data,
                  BYTE                  Msg.Length,
                  []BYTE                 Message,
                  BYTE                   SCSI.Status,
                  INT16                  Execution.Status)
```

(C):

```
int scsi_read_10( channel      *to_scsi_ha,
                  channel      *from_scsi_ha,
                  char          target_id,
                  char          lun,
                  char          dpo,
                  char          fua,
                  char          reladr,
                  int           logical_block_address,
                  int           number_of_blocks,
                  int           block_size,
                  char          *control_byte,
                  char          rx_data[],
                  char          *msg_length,
                  char          message[],
                  char          *scsi_status)
```

A call to the above procedure/function will build and issue a SCSI Read 10 command to the specified Target.ID and Logical Unit (LUN) via HADDIF, IMS B422 Device Driver and IMS B422 SCSI TRAM. Data read from the disk starting at Logical.Block.Address for Number.of.Blocks will be written into the supplied data area Rx.Data. The total number of bytes read from the disk will be Number.of.Blocks * Block.Size bytes.

If successful, an Execution.Status of SCSI.E.Good will be returned.

If Execution.Status is returned as SCSI.E.Bad.SCSI.Status, then the caller should inspect SCSI.Status, Msg.Length and Message, which are directly returned by the target peripheral device. It will probably be necessary to issue a Request.Sense command to the target peripheral device subsequently to this error condition, in order to ascertain the precise nature of the error condition and to clear the error condition.

Other returned values of Execution.Status are specific to the operation of IMS B422 and its device driver.

10.2.4 Target Mode Interface.

The Target mode interfaces provide transputer systems acting as peripheral devices, the ability to receive and execute SCSI Command Sequences supplied by initiator devices on the SCSI Bus. Interfaces are provided to initially receive a SCSI Command and for each SCSI Bus phase.

The use of target mode interfaces requires that the caller has a thorough working knowledge of the SCSI specification and the operation of peripheral devices

10.2.5 Diagnostic tests

The diagnostic tests provided, test the SCSI controller interface circuitry and the memory data areas used for passing data to and from the SCSI Bus and transputer links of IMS B422. It is a confidence test, rather than an exhaustive analysis of the hardware of IMS B422. The tests performed are as follows :-

- Test that each location in the supplied data area "Double.Buffer" may be correctly set to the hexadecimal patterns FFFF, AAAA, 5555 & 0000.
- Check for addressing errors within the supplied data area "Double.Buffer"
- Test that the Transfer.Count register within the SCSI Interface controller may be set correctly to all possible values.
- Test that the depth of the FIFO within the SCSI Interface controller is 16.
- Check that each location within the FIFO of the SCSI Interface controller may be correctly set to the hexadecimal patterns FF, AA, 55 & 00.

10.2.6 Initialisation Interfaces

The Initialisation interface, permits initiator or target applications the ability to define operating characteristics of the IMS B422 TRAM and whether to perform diagnostic tests.

The speed of data transfer across the SCSI bus may also be selected in order to facilitate the matching of data transfer speeds for slower SCSI devices.

10.3 Incorporation into a user program

Incorporation of initiator mode into a user program

The processor classes supported by the provided interface libraries (B_ALLDEV, I_ALLDEV, B_DIRTAC, I_DIRTAC, SCSICOMS and TGTMODE) are T8 and T2. Each class is compiled in halt, stop and universal modes. Vector space is disabled.

Briefly the supplied interface libraries provide the following facilities:

- SCSICOMS provides generic interfaces.
- B_ALLDEV provides modules to build byte packed SCSI command sequences for 'All Device Types', from non byte packed input parameters. It does not issue the output SCSI command sequence to IMS B422.
- I_ALLDEV provides modules to both build and then issue to IMS B422, byte packed SCSI command sequences, for 'All Device Types', from non byte packed input parameters.
- B_DIRTAC provides modules to build byte packed SCSI command sequences for 'Direct Access Device Types', from non byte packed input parameters. It does not issue the output SCSI command sequence to IMS B422.
- I_DIRTAC provides modules to both build and then issue to IMS B422, byte packed SCSI command sequences, for 'Direct Access Device Types', from non byte packed input parameters.

In general for initiator mode programs, it will be necessary to use the libraries I_ALLDEV, I_DIRTAC and SCSICOMS.

The libraries B_ALLDEV and B_DIRTAC will only be required to be used if it is necessary to have a list of already built SCSI command sequences that may then be subsequently, transferred to the generic initiator mode SCSI command interface (HADDIF).

10.4 Simple Initiator mode example.

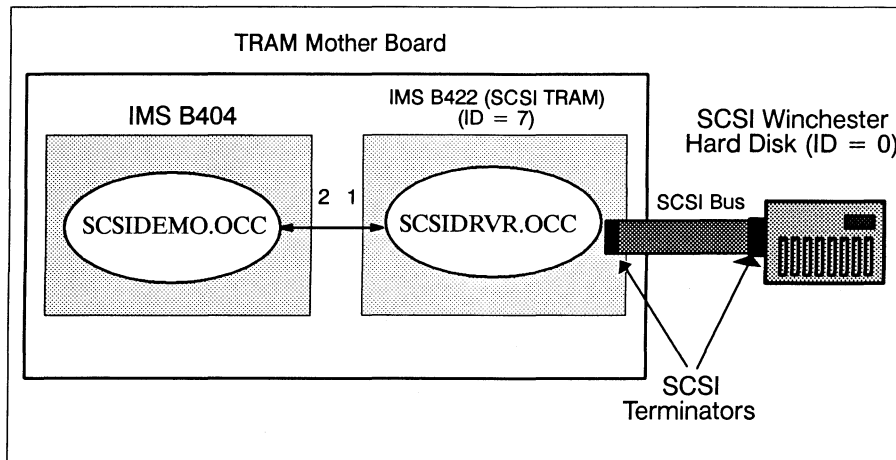


Figure 10.3 Configuration of SCSIDEMO.BTL

Configuration

```
-- Environment

-- IMS B008 : IMSB404 (slot 0), IMSB422 (slot 1).
--           Does not use C004.
-- SCSI Direct Access Device configured as Target ID 0
-- on the SCSI Bus connected to IMS B422.

-- SET ISEARCH=c:\itools\libs\ c:\imsf002a\

-- SCSIDEMO.PGM Configuration file

#include "linkaddr.inc"
#include "scsicons.inc"
#include "scsipcol.inc"

#use "scsidemo.cah"
#use "scsidrvr.c2h"

CHAN OF TO.SCSI TO.SCSI.HA:
CHAN OF FROM.SCSI FROM.SCSI.HA:

PLACED PAR
PROCESSOR 0 TA
  PLACE TO.SCSI.HA AT link2.out:
  PLACE FROM.SCSI.HA AT link2.in:
  scsi.demo (TO.SCSI.HA,
            FROM.SCSI.HA)

PROCESSOR 1 T212
  PLACE TO.SCSI.HA AT link1.in:
  PLACE FROM.SCSI.HA AT link1.out:
  INMOS.SCSI.Driver() -- Host Adaptor Device Driver
```

SCSIDEMO.OCC

```

#include "scsi.inc" --General SCSI definitions

PROC scsi.demo ( CHAN OF TO.SCSI TO.SCSI.HA,
                 CHAN OF FROM.SCSI FROM.SCSI.HA)

    #USE "scsicoms.lib"
    #USE "i_alldev.lib"
    #USE "i_dirtac.lib"

    VAL BYTE Initiator.ID IS 7 (BYTE):
    VAL BYTE Target.ID IS 0 (BYTE):
    VAL BYTE Inquiry.Data.Length IS #24 (BYTE):
    VAL BYTE Request.Sense.Data.Length IS 18 (BYTE):

    VAL INT Buff.Size IS 512:
    VAL INT32 Block.Size IS INT32 Buff.Size:
    VAL INT32 Number.of.Blocks IS 1 (INT32):

    [Buff.Size]BYTE Block:
    [255]BYTE Message:
    [Inquiry.Data.Length]BYTE Inquiry.Data:
    [Request.Sense.Data.Length]BYTE Request.Sense.Data:
    BYTE SCSI.Status, Msg.Length, :
    INT16 Execution.Status:
    INT32 Number.of.Blocks.32, Logical.Block.Address:

SEQ

    --Initialise Controller and Reset SCSI Bus

    SCSI.Initialise ( TO.SCSI.HA,
                    FROM.SCSI.HA,
                    INT16 Default.Data.Phase.Time.Out,
                    INT16 Default.Interrupt.Time.Out,
                    Enable.Parity.Checking,
                    Enable.Parity.Generation,
                    Fast.Cable.Mode,
                    Reset.SCSI.Controller,
                    Reset.SCSI.Bus,
                    Do.Self.Test,
                    Execution.Status)

    ... Check Execution Status

    Enable.Initiator.Mode( TO.SCSI.HA,
                          FROM.SCSI.HA,
                          Initiator.ID,
                          Execution.Status)

    ... Check Execution Status

    SCSI.Define.Data.Transfer.Mode ( TO.SCSI.HA,
                                    FROM.SCSI.HA,

```

```
SCSI.Fast.DMA.Mode,
Execution.Status)
```

```
--After a SCSI Bus reset, the target device will
--return with a SCSI Status check condition.
--Use the following Request Sense Command to
--Clear the check condition.
Msg.Length := Zero.B
```

```
SCSI.Request.Sense.6( TO.SCSI.HA,
FROM.SCSI.HA,
Target.ID,
LUN,
Request.Sense.Data.Length,
Non.Link.Command,
Request.Sense.Data,
Msg.Length,
Message,
SCSI.Status,
Execution.Status)
```

```
... Check Execution Status
```

```
--Should return with Good SCSI Status.
```

```
Msg.Length := Zero.B
```

```
SCSI.Inquiry.6( TO.SCSI.HA,
FROM.SCSI.HA,
Target.ID,
LUN,
Zero.B,           -- EVDP
Zero.B,           -- Page Code
Inquiry.Data.Length,
Non.Link.Command,
Inquiry.Data,
Msg.Length,
Message,
SCSI.Status,
Execution.Status)
```

```
... Check Execution Status, and act as appropriate
```

```
--Interrogate Inquiry data to find out
--about the device.
```

```
--Define simple test pattern
```

```
SEQ i = 0 FOR Buff.Size
Block[i] := BYTE (i / 2)
```

```
--Now write a single 512 byte block to the device
--at logical block 0
```

```
Logical.Block.Address := 0 (INT32)
```

```
SCSI.Write.10(TO.SCSI.HA,
```

```

        FROM.SCSI.HA
        Target.ID,
        LUN,
        DPO,
        FUA,
        RelAdr,
        Logical.Block.Address,
        Number.of.Blocks.32,
        Block.Size,
        Non.Link.Command,
        Block,
        Msg.Length,
        Message,
        SCSI.Status,
        Execution.Status)

... Check Execution Status, and act as appropriate

SEQ i = 0 FOR Buff.Size
    Block[i] := Zero.B           -- Clear buffer

SCSI.Read.10( TO.SCSI.HA,
              FROM.SCSI.HA,
              Target.ID,
              LUN,
              DPO,
              FUA,
              RelAdr,
              Logical.Block.Address,
              Number.of.Blocks.32,
              Block.Size,
              Non.Link.Command,
              Block,
              Msg.Length,
              Message,
              SCSI.Status,
              Execution.Status)

... Check Execution Status

--Check if data read = data written

SEQ i = 0 FOR Buff.Size
    IF
        (Block[i] <> (BYTE (i / 2)))
            CAUSEERROR()
    TRUE
    SKIP
:

```

Compiling, linking and running:

```
imakef scsidemo.btl
make -f scsidemo
iserver /sb scsidemo.btl /se
```

General comments:

Check the SCSI devices manual to ascertain that the SCSI command you wish the device to execute is implemented by the device, and that all the features of the command you wish to use have been implemented.

In general, use 10 byte commands in preference to 6 byte commands if your target device supports both.

10.5 List of supplied procedures.

SCSI Commands for All Device Types	SCSI Interface control procedures.	SCSI Commands for Direct Access Device Type
Change Definition 10 Compare 10 Copy 6 Copy And Verify 10 Inquiry 6 Log Select 10 Log Sense 10 Mode Select 6 Mode Select 10 Mode Sense 6 Mode Sense 10 Read Buffer 10 Rx Diagnostic Results 6 Request Sense 6 Send Diagnostic 6 Test Unit Ready 6 Write Buffer 10	SCSI Initialise Enable Target Mode Enable Initiator Mode Define Data Transfer Mode Host Adaptor Device Driver Analyse Subsystem Reset Subsystem Read Subsystem not Error	Format 6 Lock Unlock Cache 10 Pre Fetch 10 Prevent Allow Medium Removal 6 Read 6 Read 10 Read Capacity 10 Read Defect Data 10 Read Long 10 Reassign Blocks 6 Release 6 Reserve 6 Rezero Unit 6
	Target Mode Interfaces Selection Disconnect From Bus Reconnect to Bus Data In Phase Data Out Phase Status and Message In Phase Status Phase Message In Phase Message Out Phase	Search Data 10 Seek 6 and Seek 10 Set Limits 10 Start Stop Unit 6 Synchronize Cache 10 Verify 10 Write 6 Write 10 Write and Verify 10 Write Long 10
Note that the subscript numbers 6 and 10 refer to the length of the SCSI command in bytes.		

10.6 Performance

The following *sustained* transfer rates in Mbytes/second have been measured on a limited range of disk drives during the transfer of 1MByte data structures to and from the device, using an IMS B404-3 (20MHz) TRAM to control the IMS B422 SCSI TRAM in initiator mode:

Drive	Capacity	Read	Write
CDC Sabre EMD 97201	(1.2 GByte)	1.26	1.45
Maxtor XT8380S	(380 Mbyte)	1.24	1.41
Maxtor LXT200S	(200 Mbyte)	1.02	1.02
Seagate ST125N	(20 Mbyte)	0.49	0.55

Performance of SCSI drives is very dependent upon the transfer size used i.e. how many blocks are transferred by a single read or write command. The fewer blocks that are transferred, the greater the targets command overhead and rotational latency impacts upon the transfer rate. Therefore to achieve high transfer rates, request large sequential data transfers. Note that some drives, have variable geometry recording surfaces and so the transfer rate decreases as the logical block number increases.

10.7 Compatibility.

The following list of disk drives have been found to be compatible with IMS F002A running on an IMS B422 SCSI TRAM.

8"	CDC Sabre 1.2 GByte.	
5.25" Full Height	Maxtor XT 8380S	*
5.25" Half Height	Seagate ST296N	
	Seagate Wren ST2383N	*
3.5"	Maxtor LXT 200S	*
	Seagate ST125N	

* Do not use SCSI.Test.Unit.Ready.6() after resetting SCSI Bus.
Use SCSI.Request.Sense.6 to clear unit attention condition.

The following list of Seagate disk drives have been reported to work satisfactorily by customers using IMS F002B running on an IMS B422 SCSI TRAM. No performance figures are available.

8"	ST8368N	5.25" Half Height	ST2106N
	ST8500N		ST2125N
	ST8741N		ST2209N
	ST8851N		ST2383N
	ST81236N		ST2502
5.25" Full Height	ST41520N	3.5"	ST1126N
	ST4182N		ST1162N
	ST4350N		ST1201N
	ST4702N		ST1201NS
	ST4766N		ST1239NS
	ST4767N		
ST41200N			

The following host bus adaptors have been tested with IMS B422 in target mode and found to be compatible.

IBM PC	Seagate ST01 Host Bus Adaptor (16k & 8k variants). Adaptec 1540 Host Bus Adaptor.
--------	--

10.8 Operating environment

IMS F002A software is supplied in binary form and is compatible with IMS Dx205 OCCam 2 and IMS Dx214 ANSI C toolset products. An IMS B422 SCSI TRAM, a general purpose compute TRAM (e.g. IMS B404) both mounted on a suitable mother board and a SCSI device (Winchester hard disk) along with a SCSI cable, are required for program execution (figure 10.4).

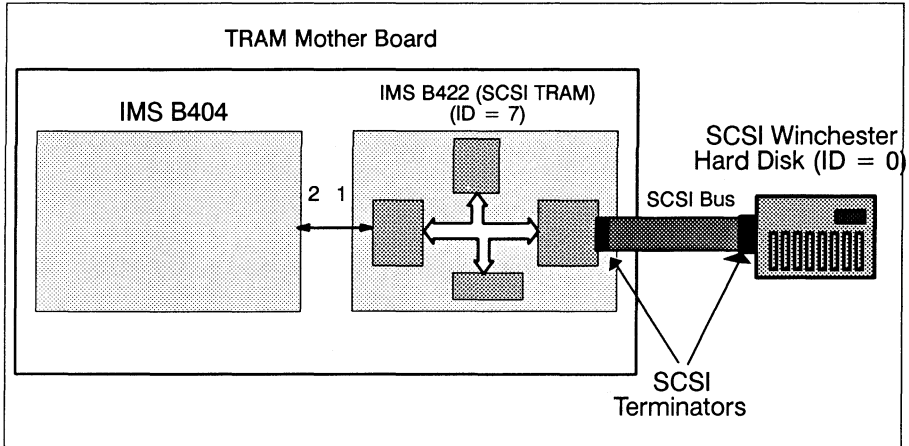


Figure 10.4 Example SCSI configuration.

For program execution, the user will require:

- An IMS B422 TRAM
- A general purpose compute TRAM, such as the IMS B404
- A suitable TRAM motherboard on which to mount the above
- A suitable SCSI cable

10.9 IMS F002B Product Components

10.9.1 Distribution media

The IMS F002B software is distributed on two media systems:

- 360Kbyte 5.25 inch IBM format diskettes
- 720Kbyte 3.5 inch IBM format diskettes

10.9.2 Documentation

- Delivery manual
- User manual

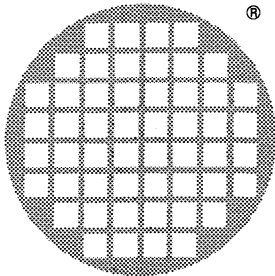
10.10 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates. Software problem report forms are included with the software. INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.

INMOS has a policy of continuously upgrading products, please contact your local distributor for the latest specification.

10.11 References

- 1 IMS B422 SCSI TRAM datasheet, INMOS Limited, September 1990
(INMOS document number 42 1273 00)
- 2 OCCaM Toolset datasheet, INMOS Limited, October 1990
(INMOS document number 42 1483 00)
- 3 ANSI C Toolset datasheet, INMOS Limited, August 1990
(INMOS document number 42 1471 00)
- 4 ANSI X3.131-1986 (SCSI-1)
- 5 ANSI X3.131-198X (SCSI-2 Working draft proposal X3T9.2/86-109 Rev.8 X3T9/89-042) (or later).



inmos®

IMS F003A

2D Graphics libraries

Product Information

A software support package for the IMS B419 Graphics TRAM

KEY FEATURES

- Functionally conformant subset of the CGI (ISO TC97/SC21 N1179) standard
- Separates CGI functions from hardware specific details
- Usable with the IMS B419 Graphics TRAM
- Compatible with INMOS ANSI C (IMS Dx214)

APPLICATIONS

- Graphical User Interfaces
- Scientific data presentation
- Engineering and interactive drawing packages
- Computer Aided Design
- Computer Animation

11.1 Introduction

The IMS F003 2D Graphics libraries offers a functionally conformant subset of the CGI (ISO TC97/SC21 N1179) standard. It is implemented in ANSI C, and is supplied with suitable C language bindings.

IMS F003 comprises two libraries, one of which implements the CGI conformant routines, while the other performs IMS B419 hardware dependent tasks. A user who wishes to use the Graphics routines with other hardware need only link in an alternative hardware library. The libraries are called `CGILIB.LIB` and `B419.LIB` respectively.

This document is a brief introduction to the features provided by IMS F003 and explains by way of example, how the product may be used.

It is assumed that the reader is familiar with:

- Transputers and C
- The CGI graphics standard
- The D7214 ANSI C toolset

11.2 Product Overview

The routines provided in the CGI library fall into 2 groups, namely Graphical Primitive functions and Graphical Attribute functions.

Graphical Primitive functions define the geometric components of a picture, and fall into the following categories:

- Line
- Marker
- Text
- Filled Area
- Image
- Generalized Drawing Primitive (GDP)

Graphical Attribute functions determine the appearance of the Graphical Primitive functions, such as line type, width and colour, character spacing, fill colour and so forth.

The user can specify 3 drawing modes, which are:

- Plot mode
- Filling mode
- Logical mode

The Plot mode defines whether or not pixels or Pels (user defined 'fat' pixels) are used as the fundamental image element.

The Filling mode defines whether a solid colour or a user designed pattern is used when region fills are performed.

The Logical mode defines whether or not logical operations are performed when image elements are plotted. For example, it is possible to perform a logical AND, OR, XOR, NAND, or NOR between the image element to be plotted and the image element in the screen memory.

All graphics operations are performed on a screen, which is a data structure representing an image. Users may define multiple screens, which may be used to perform animation or for implementation of a windowing system, for example.

11.2.1 Summary of the IMS F003 commands

The following commands comprise the `CGILIB.LIB` library. They are hardware independent, and use routines from the `B419.LIB` library to interface with the IMS B419 Graphics TRAM.

- `cgi_addsptext` - Append text at current position
- `cgi_addtext` - Append text to current text position
- `cgi_arc` - Outline part of axis-aligned ellipsoid
- `cgi_arcc` - As `cgi_arc`, with chord or segment lines
- `cgi_chrbegin` - Set current display position
- `cgi_chrspace` - Set current inter-character spacing
- `cgi_chrz` - Plot character with scaling
- `cgi_circle` - Outline an axis-aligned ellipsoid
- `cgi_cls` - Optimized screen clear
- `cgi_copy` - 2D block copy
- `cgi_disjpolyline` - Plot a series of disjoint lines
- `cgi_dot` - Plot a point
- `cgi_errstat` - Expound the current CGI error
- `cgi_fcircle` - Draw a filled axis aligned ellipsoid
- `cgi_ffan` - Draw a filled partial ellipsoid
- `cgi_fhline` - Fill a list of horizontal lline segments
- `cgi_frect` - Draw a filled rectangle
- `cgi_ftrap` - Fill a trapezoid
- `cgi_init` - Initialize the CGI library static variables
- `cgi_line` - Draw a straight line between two end points
- `cgi_paint` - Paint an existing area
- `cgi_polygon` - Plot a polygon outline
- `cgi_polyline` - Plot a polyline through coordinate points
- `cgi_rect` - Draw an axis-aligned rectangle outline
- `cgi_rot` - 2D block rotation
- `cgi_search` - Scan horizontal line segment for colour changes
- `cgi_setbcol` - Select current background drawing colour

- `cgi_setdrawmode` - Set Plot, Filling and Logical modes
- `cgi_setdrawscreen` - Select current drawing screen
- `cgi_setfcol` - Select current foreground drawing colour
- `cgi_setfillstyle` - Define a fill pattern
- `cgi_setfont` - Define a font for text display
- `cgi_setlinestyle` - Define a custom line design
- `cgi_setorient` - Select orientation for text and image copy
- `cgi_setpelstyle` - Define a custom Pel design
- `cgi_sptext` - Plot text at specified position
- `cgi_strokearc` - Outline an arc without using symmetry techniques
- `cgi_text` - Display text at specified coordinates
- `cgi_zoom` - Arbitrary XY scaling of a 2D block

The following routines comprise the `B419.LIB`

- `fs_displaybank` - Select current screen for display
- `fs_initscreen` - Assign an identifying number to a screen
- `fs_initVTG` - Initialise G300 parameters
- `fs_screenaddr` - return the field address of a screen
- `fs_setpalette` - Set up the G300 CLUT entries

The following table shows the correspondence between CGI and IMS F003 Graphical Primitive functions.

CGI	IMS F003
POLYLINE	cgi_polyline
DISJOINT POLYLINE	cgi_disjpolyline
CIRCULAR ARC CENTRE	cgi_arc
ELLIPTICAL ARC	cgi_arc
POLY MARKER	cgi_dot, cgi_copy
CELL ARRAY	cgi_frect, cgi_ftrap, cgi_copy
POLYGON	cgi_polygon, cgi_ftrap, cgi_fline, cgi_paint, cgi_search
POLYGON SET	cgi_polyline, cgi_disjpolyline, cgi_line, cgi_ftrap, cgi_fline
RECTANGLE	cgi_rect, cgi_frect
CIRCLE	cgi_circle, cgi_fcircle
CIRCULAR ARC 3 POINT CLOSE	cgi_arcc, cgi_strokearc, cgi_ffan
CIRCULAR ARC CENTRE CLOSE	cgi_arcc, cgi_strokearc, cgi_ffan
ELLIPSE	cgi_circle, cgi_fcircle, cgi_strokearc
ELLIPTICAL ARC CLOSE	cgi_arcc, cgi_strokearc, cgi_ffan
TEXT	cgi_text, cgi_sptext, cgi_chr, cgi_chrbegin, cgi_chrspace
APPEND TEXT	cgi_addtext, cgi_addsptext, cgi_chr, cgi_chrspace
RESTRICTED TEXT	cgi_text, cgi_sptext, cgi_chr, cgi_chrbegin, cgi_chrspace

Table 11.1 Correspondence between CGI and IMS F003 Graphical Primitive Functions

The following table shows the correspondence between CGI and IMS F003 Attribute functions.

CGI	IMS F003
LINE TYPE	cgi_setlinestyle, cgi_setdrawmode
LINE WIDTH	cgi_setlinestyle, cgi_setdrawmode
LINE COLOUR	cgi_setlinestyle, cgi_setdrawmode
MARKER TYPE	cgi_setpelstyle, cgi_copy, cgi_zoom
MARKER SIZE	cgi_setpelstyle, cgi_copy, cgi_zoom
MARKER COLOUR	cgi_setpelstyle, cgi_copy, cgi_zoom
TEXT FONT INDEX	cgi_setfont
TEXT PRECISION	cgi_text, cgi_chr, cgi_zoom
CHARACTER EXPANSION FACTOR	cgi_chr, cgi_zoom
CHARACTER SPACING	cgi_chrspace
TEXT COLOUR	cgi_setfcol
CHARACTER HEIGHT	cgi_chr
CHARACTER ORIENTATION	cgi_setorient, cgi_rot
CHARACTER SET INDEX	cgi_setfont
ALTERNATE CHARACTER SET INDEX	cgi_setfont
INTERIOR STYLE	cgi_setfillstyle, cgi_setfcol
FILL COLOUR	cgi_setfcol
HATCH INDEX	cgi_setfillstyle
PATTERN INDEX	cgi_setfillstyle
PATTERN TABLE	cgi_setfillstyle
PATTERN SIZE	cgi_setfillstyle

Table 11.2 Correspondence between CGI and IMS F003 Attribute functions

11.3 Using IMS F003

This section demonstrates the usage of IMS F003 by way of example. The example is supplied in source on the release diskette. It initialises the IMS B419 TRAM using B419.LIB routines and displays an INMOS wafer logo using routines from CGILIB.LIB.

The C function prototypes for both libraries are referenced via the `cgidefs.h` header file, which also provides various important constant definitions.

The CGILIB.LIB and B419.LIB libraries must be linked with the code when building a bootable image. This is done using the standard INMOS link utility, `ilink`.

```
#include <channel.h>
#include <stdio.h>
#include <mathf.h>
#include <math.h>
#include <stdlib.h>

#include "cgidefs.h"

void palette ()
{
    int clutloc=0;
    int colour=0;

    for (clutloc=0; clutloc<64; clutloc++)
    {
        colour = clutloc*4;
        fs_setpalette (clutloc,colour,colour,colour); /* grey scale */
        fs_setpalette (clutloc+64 ,colour,0,0); /* red scale */
        fs_setpalette (clutloc+128,0,colour,0); /* green scale */
        fs_setpalette (clutloc+192,0,0,colour); /* blue scale */
    }
}

int boxThere (x,y)
int x,y;
{
    static int xMissing[] = { 0, 1, 0, 0, 1, 0, 7, 7 } ;
    static int yMissing[] = { 0, 0, 1, 7, 7, 6, 6, 7 } ;
    int i;

    for (i=0;i<8;i++)
        if (((xMissing [i] == x) && (yMissing [i] == y)) ||
            ((yMissing [i] == x) && (xMissing [i] == y)))
            return (0);
    return (1);
}

void logo (x0,y0,rad,fg_col,bg_col)
int x0,y0; /* top left coords */
int rad; /* radius */
int fg_col,bg_col; /* box, circle colours */
{
    int boxEdge,holeEdge;
    int xc, yc ;
    int xt, yt ;
}
```

```

boxEdge=rad/5;
holeEdge=rad/39;
xc = (x0 + rad) + 9;
yc = (y0 + rad) + 9;
cgi_setfcol(bg_col);
cgi_fcircle (xc,yc,rad,rad);
{
  int x0 = (xc - (boxEdge << 2 )) - ((holeEdge * 7) / 2) ;
  int y0 = (yc - (boxEdge << 2 )) - ((holeEdge * 7) / 2) ;
  int x,y;

  for (x=0;x<8;x++)
    for (y=0;y<8;y++)
      if (boxThere (x,y))
        {
          xt= x0 + ((boxEdge+holeEdge) * x);
          yt= y0 + ((boxEdge+holeEdge) * y);
          cgi_setfcol(fg_col);
          cgi_frect (xt,yt,xt+boxEdge,yt+boxEdge);
        }
}

screen screens [2];
int visible_screen, invisible_screen;

/* These parms are suitable for 1024 by 1024
   on 50-60kHz linescan monitor */

int g3parms[] = {17,17,43,256,87,164,6,56,2048,338,0,491,21,255};

int main ()
{
  int loop1,loop2;
  char text1[] = {"inmos ltd"};
  char text2[] = {"IMS F003 C Graphics Libraries"};

  cgi_init ();
  fs_initVTG (g3parms);
  visible_screen = 0;
  invisible_screen = 1;
  fs_initscreen (&screens [visible_screen ],visible_screen ,1024,1024);
  fs_initscreen (&screens [invisible_screen],invisible_screen,1024,1024);
  fs_displaybank (visible_screen);
  palette ();

  cgi_setdrawscreen (&screens[visible_screen]);
  cgi_setfcol (127);
  cgi_cls (&screens [visible_screen],255);
  cgi_cls (&screens [invisible_screen],255);
  for (loop1=0;loop1<256;loop1++)
  {
    cgi_setfcol (loop1);
    for (loop2=0;loop2<4;loop2++)
      cgi_line ((loop1*4)+loop2,0,(loop1*4)+loop2,1023);
  }
  cgi_setdrawmode (PM_COL, RM_XOR, FM_COL);
  for (loop1=0;loop1<256;loop1++)

```

```
{
  cgi_setfcol (loop1);
  for (loop2=0;loop2<4;loop2++)
    cgi_line (0,(loop1*4)+loop2,1023,(loop1*4)+loop2);
}
cgi_setdrawmode (PM_COL, RM_COL, FM_COL);
for (loop1=0;loop1<64;loop1++)
{
  cgi_setfcol (loop1);
  for (loop2=0;loop2<8;loop2++)
    cgi_line (256,256+(loop1*8)+loop2,768,256+(loop1*8)+loop2);
}
cgi_setfcol (63);
cgi_rect (256,256,768,768);
logo (270,390,100,40,20);
cgi_setfont (font8by8, 8, 2, 4);
cgi_setorient (TX_NORM);
cgi_chrspace (10,0);
cgi_setfcol (127);
cgi_setdrawmode (PM_COL, RM_XOR, FM_COL);
for (loop1=0;loop1<9;loop1++)
{
  cgi_chrbegin (518+(loop1*23),497);
  cgi_chrz (text1[loop1],20,20);
}
for (loop1=0;loop1<32;loop1++)
{
  cgi_chrbegin (290+(loop1*14),700);
  cgi_chrz (text2[loop1],12,14);
}
}
```

11.4 IMS F003 Product Components

11.4.1 Distribution media

The IMS F003 software is distributed on two media systems:

- 360Kbyte 5.25 inch IBM format diskettes
- 720Kbyte 3.5 inch IBM format diskettes

11.4.2 Documentation

- Delivery manual
- User manual

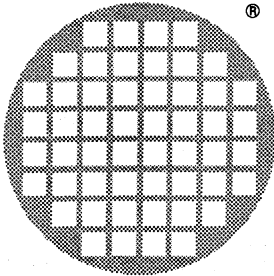
11.4.3 Operating environment

For program execution, the user will require:

- An IMS B419 Graphics TRAM
- A suitable TRAM motherboard on which to mount the above
- A colour monitor with RGB Video input

11.5 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates. Software problem report forms are included with the software. INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.



inmos[®]

IMS F007A

DSP libraries, C interface generator & linker

Product Information

Software for the IMS B420 VecTRAM

KEY FEATURES

- A library of more than a 100 ANSI C routines for common signal/vector processing applications.
- Compatible with Zoran's library (**ZoranLib 2.0**) for ZR34325.
- ANSI C toolset compatible (IMS Dx214).
- Dramatically speeds up parallel applications and increases system performance involving vector/signal processing computation.
- When a vector library function is encountered, the co-processor is activated to execute the required function transparently.
- Automatically copies data structures from non shared memory areas to a shared memory area accessible by both the transputer and the Zoran ZR34325 vector co-processor.
- Obviates the need for users to directly program the hardware of IMS B420 for common vector/signal processing operations.
- The supplied C interface generator (intgen) and VecTram linker (ivtlink) binds transputer and Zoran ZR34325 programs into a single executable unit for execution on VecTRAM.
- Permits users to create their own libraries using a combination of Zoran and Inmos software development tools.

APPLICATIONS

- Speech processing.
- Communication and coding.
- Graphics and numerical processing.
- Radar, sonar, ultrasonics, etc.
- Seismic/geophysical data processing.
- Neural networks.
- Image processing and compression.

12.1 Introduction

The IMS F007A consists of a library of C functions developed specifically for common vector/signal processing tasks, encountered in many applications. The functions are callable from a C program running on an IMS B420 TRAM, and can be used to dramatically speed up parallel applications and system performance involving vector/signal processing computation.

In addition to the comprehensive library, tools are provided that permit users to create their own application specific libraries of VecTRAM routines. The tools permit transputer C programs, written for the IMS B420 VecTRAM, and Zoran Assembler programs (ASM325) for the ZR34325, to be bound into a single executable unit.

The libraries and development tools that are provided are as follows :

Libraries

- **ivtlib.h, ivtlib.lib and ivttool.lib**, C interface libraries for **ZoranLib** application library.
- **ivtlib.libz**, a modified version of the ZR34325 **ZoranLib** application library.

Development Tools

- **Intgen**, an interface generation utility which can automatically generate C interface functions for new ZR34325 subroutines.
- **Ivtlink**, a VecTRAM Linker which links Transputer object and library files with ZR34325 object and library files.

This document is a brief introduction to the features provided by IMS F007A and explains, by way of example, how the product may be used.

It is assumed that the reader is familiar with :

- Transputers and C
- Digital Signal Processing (DSP)
- IMS Dx214 ANSI C toolset [1]

12.2 Product Overview

12.2.1 Library usage.

During program execution, when a vector library function is encountered, the co-processor is activated to execute the required function. On the IMS B420 TRAM, the co-processor has its own local fast memory space, which is also accessible to the transputer. However the transputer local memory is not visible to the co-processor. When a library function is called; if the data to be processed is in the transputer local memory space, it is automatically copied (using the block move capability of the T800) to a predetermined area in the co-processor space. The co-processor is then activated to execute the required function.

If the destination vector operand address, specified in the call, is in the transputer space, the processed data is automatically copied back to the specified area in the transputer memory space. This built-in copying means that the co-processor operation can be totally transparent to the programmer, and programs can be accelerated without the need for detailed knowledge about the operation of the IMS B420 TRAM.

The overhead associated with data copying, between the transputer and co-processor (or visa versa), is avoided, if the source and/or destination operands for the specified function are already in the

co-processor local memory space. The library functions automatically check the operand addresses and take appropriate action. In general, if the address of an input or an output operand, in a function call, is in the co-processor space, no data copying will take place for that operand. This is particularly important if operands are to undergo several vector/signal processing operations. For optimal performance, in such cases, the user can easily specify destination (and/or source) addresses which are local to the co-processor address space. In this way data copying, between the transputer and the shared memory area, can be minimised.

Apart from vector and arithmetic functions, the IMS F007A includes efficient vector move functions which allow optimisation at the application level. The library also supports co-processor control calls which are used to set rounding modes, etc.

12.2.2 *intgen* - VecTRAM Interface Generation Utility

To use any new user-written ZR34325 subroutines in a transputer C program, one must prepare a C interface function for every ZR34325 subroutine. This task is tedious and error prone. The *intgen* utility automates this process by generating the C interface functions from an ASM325 header file.

The idea is that the same ASM325 header file that is used in ZR34325 assembly code will be used to generate the C interface library. Therefore, any additional information required by *intgen* is put into comments ('/*' ... '*/').

The operation of the *intgen* utility in terms of input and output file extensions is shown in figure 12.1.

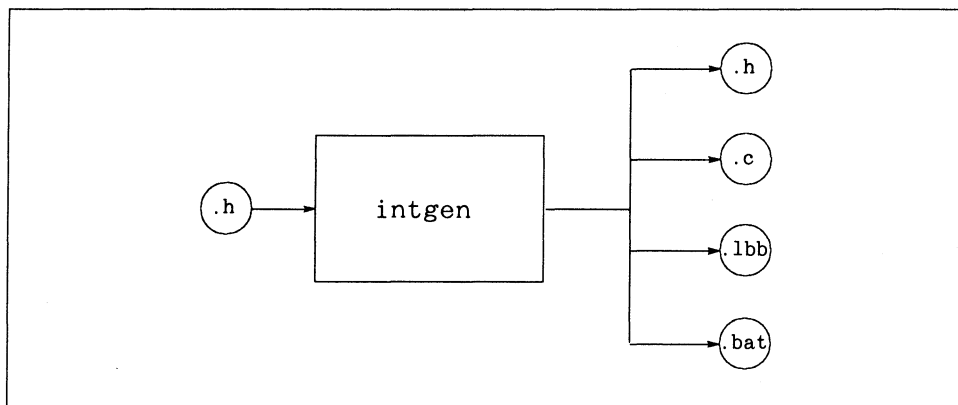


Figure 12.1 *intgen* flow diagram

Running the *intgen* utility

To invoke the *intgen* utility use the following command line:

```
intgen ASM325-header [C-header]
```

The input *ASM325-header file* includes declarations of ZR34325 subroutines with additional *intgen* information placed in comments. The output *C-header file* includes ANSI-C declarations of the interface functions.

Each interface function is generated in a separate C file, with the name of the ZR34325 subroutine. Two additional output files are a '.bat' file and a '.lbb' file which includes commands for compiling the C interface files and building the C interface library. The names of these files are given after the name of the output *C-header file*.

If the second *C-header file* is omitted from the *intgen* command line, the output ANSI C declarations are sent to the standard output. The name of the input *ASM325-header file* is used for the output '.bat' and '.lbb' files.

12.2.3 `ivtlink` – VecTRAM Linker

The VecTRAM linker tool `ivtlink`, combines a number of compiled transputer modules and libraries with assembled ZR34325 modules and libraries into a transputer linked object file for the IMS B420 VecTRAM.

The operation of the VecTRAM linker in terms of the standard input and output file extensions is shown in figure 12.2.

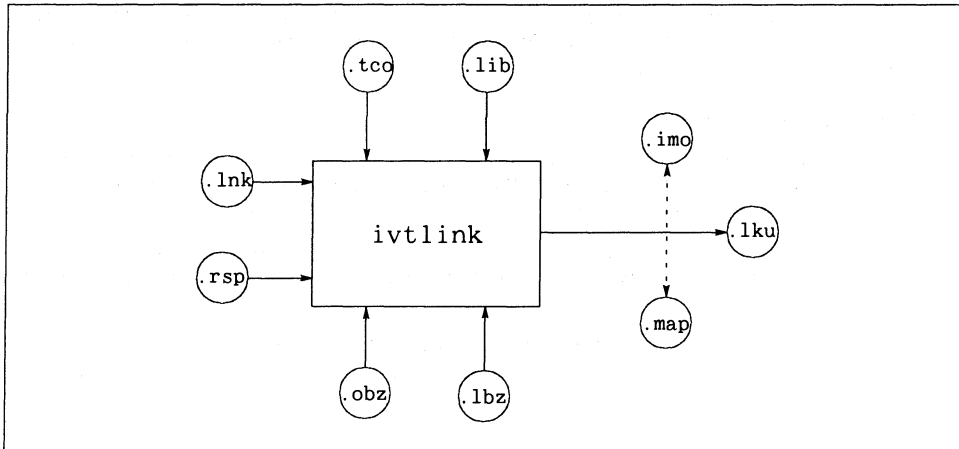


Figure 12.2 `ivtlink` flow diagram

To invoke the VecTRAM linker the following command line is used:

```
ivtlink [filenames] {options}
```

If an error occurs during the linking operation no output files are generated.

Example of use:

```
icc myprog.c /t8
ivtlink myprog.tco ivtlib.lib ivttool.lib /f startup.lnk ivtlib.lib
icollect myprog.lku /t
iserver /sb myprog.btl /se
```

In this example the compiled file `myprog.tco` is linked to the standard VecTRAM C library `ivtlib.lib` and the VecTRAM ZR34325 library `ivtlib.libz`, using the startup file `startup.lnk`. The example also shows the steps for compiling, booting and loading the program.

12.3 Supplied Routines

The following list of C functions are provided by the library `lvlib.lib` :

Name	Description
C_325_RADD	Vector Add
C_325_RADDCN	Vector Scalar Add
C_325_RATAN2	Vector Two Argument Arctangent
C_325_RCOMP	Vector Compare
C_325_RCMPCN	Vector Scalar Compare
C_325_RCOS	Vector Cosine
C_325_RDIV	Vector Divide
C_325_RDOT	Dot Product
C_325_REXP	Vector Exponential (Base e)
C_325_REXP2	Vector Exponential (Base 2)
C_325_REXP10	Vector Exponential (Base 10)
C_325_HIST	Histograms
C_325_RLIMITH	Vector Clip High Limit
C_325_RLIMITHCN	Vector Scalar Clip High Limit
C_325_RLIMITL	Vector Clip Low
C_325_RLIMITLCN	Vector Scalar Clip Low Limit
C_325_RLIMITHL	Vector Clip Low-High Limits
C_325_RLIMITLHCN	Vector Scalar Clip Low-High Limits
C_325_RLOGE	Vector Logarithm (Natural)
C_325_RLOGEZ	Vector Logarithm with Zero Check (Natural)
C_325_RLOG2	Vector Logarithm (Base 2)
C_325_RLOG2Z	Vector Logarithm with Zero Check (Base 2)
C_325_RLOG10	Vector Logarithm (Base 10)
C_325_RLOG10Z	Vector Logarithm with Zero Check (Base 10)
C_325_RMAX	Vector Maximum Element
C_325_RMIN	Vector Minimum Element
C_325_RMULT	Vector Multiply
C_325_RMULTCN	Vector Scalar Multiply
C_325_RMULTI	Vector Multiply (Integer by Float)
C_325_RMULTICN	Vector Scalar Multiply (Integer by Float)
C_325_RPOW	Vector Power
C_325_RRECIP	Vector Reciprocal
C_325_RSIGN	Vector Signum Function
C_325_RSIGN2	Two Arguments Signum Function
C_325_RSIN	Vector Sine
C_325_RSQRT	Vector Square Root

Table continued overleaf

Name	Description
C_325_RSQRTZ	Vector Square Root with Input Check
C_325_RSUB	Vector Subtract
C_325_RSUBCN	Vector Scalar Subtract

Table 12.1 Basic Real Vector Operations

Name	Description
C_325_CADD	Complex Vector Add
C_325_CADDCN	Complex Vector Scalar Add
C_325_CCOPY	Complex Vector Copy
C_325_CDIV	Complex Vector Divide
C_325_CFCONV	Complex Fast Convolution
C_325_CMULT	Complex Vector Multiply
C_325_CMULTH	Complex Vector Multiply Hermitian
C_325_CRECIP	Complex Vector Reciprocal
C_325_CSUB	Complex Vector Subtract
C_325_CSUBCN	Complex Vector Scalar Subtract

Table 12.2 Basic Complex Vector Operations

Name	Description
C_325_CFFT	Complex FFT
C_325_CFFT2K	Complex FFT 2K
C_325_CFFT4K	Complex FFT 4K
C_325_CFFT8K	Complex FFT 8K
C_325_CFFT16K	Complex FFT 16K
C_325_CFFT32K	Complex FFT 32K
C_325_CFFTI	Complex FFT (Integer Input)
C_325_CFFTI2K	Complex FFT (Integer Input) 2K
C_325_CFFTI4K	Complex FFT (Integer Input) 4K
C_325_CFFTI8K	Complex FFT (Integer Input) 8K
C_325_CFFTI16K	Complex FFT (Integer Input) 16K
C_325_CFFTI32K	Complex FFT (Integer Input) 32K
C_325_CFFTIS	Complex FFT (Integer Input) Small Sequence
C_325_CFFTS	Complex FFT Small Sequence
C_325_CIFFT2K	Complex Inverse FFT (Integer Input) 2K
C_325_CIFFT4K	Complex Inverse FFT (Integer Input) 4K
C_325_CIFFT8K	Complex Inverse FFT (Integer Input) 8K
C_325_CIFFT16K	Complex Inverse FFT (Integer Input) 16K
C_325_CIFFT32K	Complex Inverse FFT (Integer Input) 32K
C_325_CIFFTI	Complex Inverse FFT (Integer Input)

Table continued opposite

Name	Description
C_325_CIFFTI2K	Complex Inverse FFT (Integer Input) 2K
C_325_CIFFTI4K	Complex Inverse FFT (Integer Input) 4K
C_325_CIFFTI8K	Complex Inverse FFT (Integer Input) 8K
C_325_CIFFTI16K	Complex Inverse FFT (Integer Input) 16K
C_325_CIFFTI32K	Complex Inverse FFT (Integer Input) 32K
C_325_CIFFTIS	Complex Inverse FFT (Integer Input) Small Sequence
C_325_CIFFTS	Complex Inverse FFT Small Sequence
C_325_RFFT	Real FFT
C_325_RFFT1K	Real FFT 1K
C_325_RFFT2K	Real FFT 2K
C_325_RFFT4K	Real FFT 4K
C_325_RFFT8K	Real FFT 8K
C_325_RFFT16K	Real FFT 16K
C_325_RFFT32K	Real FFT 32K
C_325_RFFT256	Real FFT 256 point
C_325_RFFT512	Real FFT 512 point
C_325_RFFTI	Real FFT (Integer Input)
C_325_RFFTI1K	Real FFT (Integer Input) 1K
C_325_RFFTI2K	Real FFT (Integer Input) 2K
C_325_RFFTI4K	Real FFT (Integer Input) 4K
C_325_RFFTI8K	Real FFT (Integer Input) 8K
C_325_RFFTI16K	Real FFT (Integer Input) 16K
C_325_RFFTI32K	Real FFT (Integer Input) 32K
C_325_RFFTI256	Real FFT (Integer Input) 256 point
C_325_RFFTI512	Real FFT (Integer Input) 512 point
C_325_RFFTIS	Real FFT (Integer Input) Small Sequence
C_325_RFFTS	Real FFT Small Sequence

Table 12.3 Signal Processing And Polynomial Operations

Name	Description
C_325_CFFT2	Complex 2-D FFT
C_325_CFFTI2	Complex 2-D FFT (Integer Input)
C_325_CIFFT2	Complex 2-D Inverse FFT
C_325_CIFFTI2	Complex 2-D Inverse FFT (Integer Input)
C_325_RFFT2	Real 2-D FFT
C_325_RFFTI2	Real 2-D FFT (Integer Input)

Table 12.4 Image Processing Operations

12.4 Environment

IMS F007A software is supplied in binary form and is compatible with IMS Dx205 Occam 2 and IMS Dx214 ANSI C toolset products.

For program execution, the user will require:

- An IMS B420 VecTRAM
- A suitable TRAM motherboard on which to mount the above

12.5 IMS F007A Product Components

12.5.1 Distribution media

The IMS F007A software is distributed on two media systems:

- 360Kbyte 5.25 inch IBM format diskettes
- 720Kbyte 3.5 inch IBM format diskettes

12.5.2 Documentation

- Delivery manual
- User manual

12.6 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates. Software problem report forms are included with the software. INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.

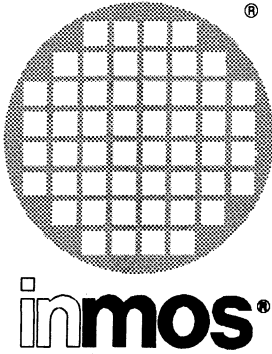
12.7 References

- 1 ANSI C Toolset datasheet, INMOS Limited, August 1990 (INMOS document number 42 1471 00)

12.8 Ordering Information

Description	Order Number
IMS F007A VecTRAM libraries and linker	IMS F007A-1

Table 12.5 Ordering Information



IMS S217
IMS S308
IMS S514
IMS S708

Device drivers &
motherboard support
software

Product Information

KEY FEATURES

- Host device drivers
- INMOS server program plus sources
- Tool for mapping transputer networks
- Support for INMOS development systems
- Support for application programs
- Support for new host machines

Host machine	Operating system	IMS B008	IMS B014	IMS B015	IMS B016	IMS B017
IBM PC XT, AT	DOS 3.0	IMS S708				
NEC PC-9801	DOS 3.0			IMS S708		
Sun 3	SunOS 4.1		IMS S514		IMS S514	
Sun 4	SunOS 4.1		IMS S514		IMS S514	
Sun 386i	SunOS 4.0.2	IMS S308				
Sun 386i	DOS 3.0	IMS S708				
IBM PS/2	DOS 3.0					IMS S217

Table 13.1 Board support software matrix

13.1 Product overview

The device driver and motherboard support software products support a wide range of motherboard and host configurations. The board software support matrix (table 13.1) defines the host machine, operating system and board combinations supported by each product.

Each board support product consists of a device driver¹, an INMOS server, software to map transputer networks and software to set the IMS C004 link switches on INMOS module motherboards.

13.1.1 Device driver

The device driver allows the motherboard to be installed as a standard device in the host machine which can be accessed via the operating system calls rather than accessing the interface registers directly.

13.1.2 INMOS server

The INMOS iserver is the standard mechanism for loading transputer binaries to transputers, and providing the transputer application with basic operating system services such as terminal input and output, file input and output, access to environment variables and so forth.

The INMOS server supports the execution of all INMOS development tools on transputers, and the execution of all applications built by the development tools.

The INMOS iserver is written C and easily ported to new development hosts. The source of the iserver is provided in each product. The iserver source can easily be extended to support new host services such as graphics. Developers may distribute both the source and binary of the iserver with their products free of charge. The device driver access within the iserver is encapsulated behind a simple interface called linkio.c. The linkio.c interface defines the set of C functions used to open, close, read from and write to the link interface.

Developers wishing to support their own boards with the iserver, simply need to reimplement the linkio.c module to call their own device driver. Having reimplemented the iserver, all INMOS development tools can now be executed on the developers own transputer hardware.

A wide range of companies support the linkio.c and iserver interfaces to their board and software products allowing easy adoption of new developments.

13.1.3 Transputer mapping software

The transputer mapping software assists the developer in checking the construction of transputer networks. The tool will list processors constituting the network and their interconnections. The tool can also be used to check the correct setting of link switches.

¹The current version of the IMS S217 software contains an implementation of the linkio.c module which directly accesses the interface board registers.

13.1.4 Switch setting support

Many of the INMOS motherboards use IMS C004 link switches to allow the developer to set the transputer network topology under software control. The module motherboard software supports the setting of these switches.

13.2 Product components summary

13.2.1 Documentation

User manual

This describes how to install the software. It describes the board installation and test procedures. It describes how to use each of the software tools provided in the package.

13.2.2 Software

Device driver

INMOS Iserver (with sources)

Transputer mapping tool

Motherboard switch-setting software

13.3 Product variants

Table 13.1 defines the host machine and operating system requirements for each of the product variants. The general operating requirements and distribution media for each product are listed below.

13.3.1 General operating requirements

Each product will require:

- 1 Mbyte of free disk space
- INMOS TRAM with at least 1 Mbyte of memory to run motherboard software
- Operating system as defined by table 13.1 (or later version)

13.3.2 Distribution media

IMS S217

Software distributed on 720 Kbyte 3.5 inch IBM format floppy disks.

IMS S308

Software distributed on 720 Kbyte 3.5 inch floppy disks in bar format.

IMS S708

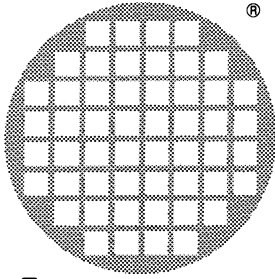
Software distributed on BOTH 360 Kbyte (48TPI) 5.25 inch IBM format floppy disks and 720 Kbyte 3.5 inch IBM format floppy disks.

IMS S514

Software distributed on DC600A data cartridges 60 Mbyte, QIC-11, in tar format.

13.4 OEM support

Customers wishing to distribute INMOS device drivers with their own products, or wishing to adapt INMOS device drivers for their own boards should contact SGS-THOMSON Sales Offices to get licensing details.



IMS S707
IMS S607
IMS S507

Network support software



Product Information

KEY FEATURES

- Transparent access to transputer networks over Ethernet
- Management of multiple transputer networks in a multi-user environment
- Support for mixed machines on host network
- Support for IMS B300 Ethernet Link Box.
- Support for IMS B018 Ethernet VME motherboard
- Fast download of programs over Ethernet to transputer networks
- Support for remote debugging of programs from host terminal
- Support for remote execution of application programs
- Support for IBM PC, VAX VMS, Sun 3, Sun 4

14.1 Introduction

The INMOS Network Support Software products assist in the management of transputers attached to computer networks, providing shared access to transputer resources in a manner transparent to the user.

The product variants supported are

IMS S707 IBM PC

IMS S607 VAX VMS

IMS S507 Sun 3 and Sun 4

This datasheet describes the model on which the products are based and the functions supported. The operating requirements for each product variant are defined at the end of the data sheet.

14.2 Product Overview

14.2.1 Connecting transputers to computer networks

Transputers may be easily connected to computer networks. The connection may be achieved by inserting an INMOS motherboard into a host machine (a B014 in a Sun 4, or a B008 in an IBM PC for example), by using the IMS B300 Ethernet Link Box or by using the IMS B018 VME Ethernet motherboard in a VME rack. Each of these approaches gives one or more 'User Links' to which transputer networks can be directly attached. Programs may be loaded to the transputer networks down the User Links and operating system services are provided to the programs by communicating with servers across these User Links.

14.2.2 Capabilities

The system administrator is able to define a collection of capability strings for each User Link to reflect the capability of the transputer network connected to that link. The system administrator must then communicate the meaning of the capability strings to the users.

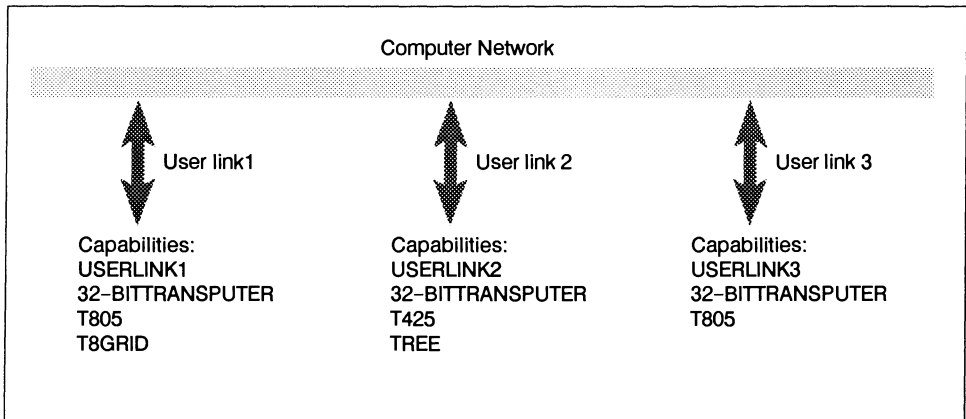


Figure 14.1 Capabilities for User Links

Figure 14.1 illustrates capabilities defined for three user links from a computer network. When a user runs a program a capability string will be specified to the INMOS iserver. The system will then search for a free User Link with a matching capability string and create the connection for the user. If a user specifies the capability string '32-BITTRANSPUTER' then he may be connected to any of User Links 1 to 3, if the user specifies the capability string 'T8GRID' he will only be connected to User Link 1, assuming of course that it is free for use at that time. System administrators should ensure that each User Link has among its capability strings a unique string so that if necessary a user may request a specific User Link to be used.

The INMOS iserver has been extended so that a user may claim a User Link for a sequence of program executions, allowing for example the post-mortem debugger to be run following a failed program execution.

14.2.3 System configuration

Each host machine has a configuration file which defines the device types and capability strings for the User Links attached to the local host machine, and a sequence of host names. When the server is invoked it will first search for a local resource to satisfy the capability request, then failing local capability or availability the remote host connection servers are interrogated in turn.

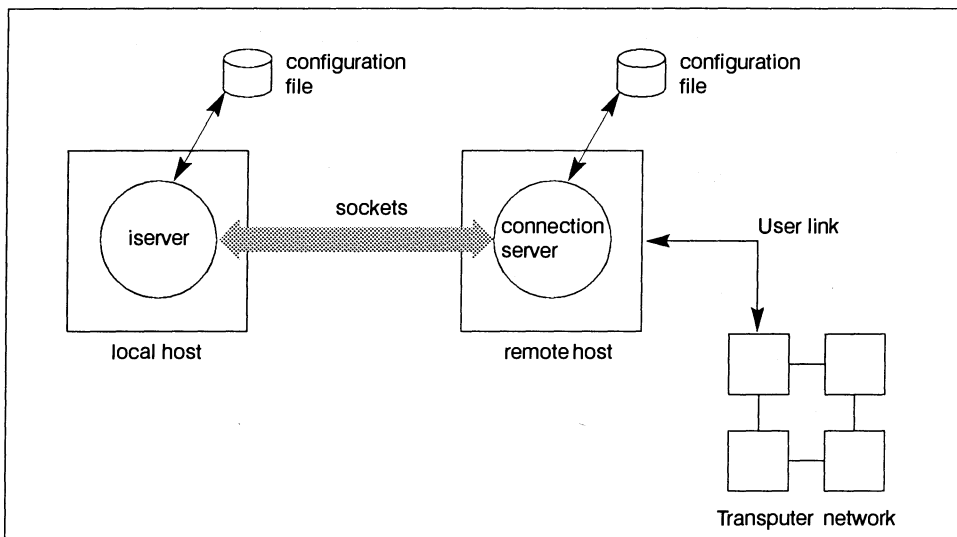


Figure 14.2 Access to transputers on remote hosts

This method of configuration description means that configuration files may be maintained local to each host machine, and that if necessary the scope of remote hosts can be prioritized and restricted.

14.2.4 Support for INMOS development tools

The INMOS development tools are fully supported by the Network Support Software. Transputer hosted tools may be run on transputer networks connected to User Links. In particular the symbolic interactive and post-mortem debugging tools may be used. Applications built with the tools are also supported.

14.2.5 Support for INMOS Ethernet connection system (IMS B300)

Tools are provided to initialise the INMOS Ethernet connection system with operating parameters (eg. Internet address) and to set up the capabilities of the User Links. Once the Ethernet connection system has been initialised it operates in exactly the same way as any other host machine. The connection services are built into the INMOS Ethernet connection system firmware.

14.2.6 Support for mixed networks of machines

The product variants for Sun 3, Sun 4, VAX VMS and PC will operate together in a mixed network. The VAX product requires a TCP implementation to be running, and the PC requires the Sun PC-NFS package.

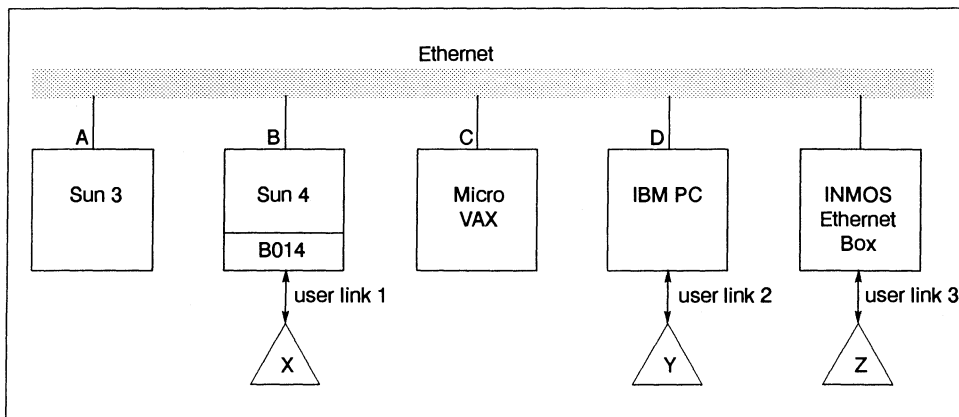


Figure 14.3 Support for mixed networks of machines

Given these requirements are met then transputer networks plugged into Sun machines can be accessed from PC, and transputer networks plugged into PCs may be accessed from Sun machines¹. The INMOS Ethernet box may be accessed from Sun 3, Sun 4, VAX VMS and PC machines. In the context of figure 14.3 it is possible to access the transputer networks X, Y and Z from the host machines A, B, C, and D.

14.3 Product Component Summary

14.3.1 Documentation

User Manual

This describes how to install the software and set up the configuration files for the users. It describes how to set up and initialise an INMOS Ethernet box. It describes how users should use the iserver to load and debug programs.

14.3.2 Software

The software product consists of

Extended INMOS Iserver

Connection server

INMOS Ethernet box configuration program

14.4 Product Variants

14.4.1 IMS S707 IBM PC

Operating requirements

- IBM PC XT or AT
- 3 Mbytes of free disk space
- DOS 3.0 or later

¹. Note that only one transputer network may be accessed on a particular PC, and that the PC must exclusively run the connection server software provided

- Sun PC-NFS or compatible software

Boards supported

- IMS B004
- IMS B008 Motherboard

Distribution media

Software is distributed on **BOTH** 360 Kbyte (48TPI) 5.25 inch IBM format floppy disks **AND** 720 Kbyte 3.5 inch IBM format floppy disks.

14.4.2 IMS S607 VAX VMS**Operating requirements**

- 3 Mbytes of free disk space
- VMS 5.2 or later
- WIN TCP/IP 5.0.2 or DEC TCP/IP

Distribution media

Software is distributed on TK50 tapes in VAX VMS backup format.

14.4.3 IMS S507 Sun 3 and Sun 4**Operating requirements**

- Sun OS 4.1 or later
- 3 Mbytes free disk space

Boards supported

- IMS B014 VME motherboard
- IMS B016 VME master board

Distribution media

Software is distributed on DC600A data cartridges 60 Mbyte, QIC-11, in tar format.

14.5 Error Reporting And Field Support

A registration form is provided with each product. Return of the registration form will ensure you are informed about future product updates.

Software problem report forms are included with the software.

INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Region Technology Centres, and authorised distributors.



Real-time Kernels and Operating Systems



VRTX32/T

Versatile Real-time Executive for Transputers



Advance Information

Features

- High performance real-time multi tasking executive for transputers (T2/T4/T8) and the next generation INMOS transputer, codenamed H1.
- Enhances the transputer hardware scheduler capabilities
- Deterministic, fixed-cost system calls, independent of the number of tasks, queues, and other system objects
- Pre-emptive priority based task scheduler
- Flexible inter-task synchronization and communication with mailboxes, queues, event flags and semaphores
- Global 32-bit event flags with overrun detection
- Counting semaphores for mutual exclusion
- Efficient time-slice option
- Reliable, high performance queues for message passing with priority or FIFO ordering
- System calls for memory allocation, real-time clock, character I/O, interrupt handling and initialisation
- Extended to provide support for interprocessor communication and distributed real-time systems.
- Fully compatible with INMOS ANSI C Compiler (IMS Dx214)
- Fully supported by RTscope, a comprehensive real-time multitasking de-bugger and VTRX32 system monitor, extended to provide distributed debugging in a multi transputer system
- Supported by Ready Systems and SGS-THOMSON field and factory support organizations

15.1 Overview

VRTX32 for transputers (VRTX32/T) is a multitasking executive designed for real-time embedded computer applications. Such applications include communications, instrumentation, military/aerospace, and factory automation. Since VRTX32 has been extensively tested, developers can begin to work on the end application immediately. There is no delay in the product development schedule while a kernel is developed and tested in-house.

VRTX32 works on transputers without modification. It does not depend on any particular implementation of interrupts, timers, memory management, buses or I/O devices. The VRTX32 implementation on transputers is fully compatible with the INMOS ANSI C compiler, thus allowing VRTX32/T applications to be written in C. VRTX32/T can be combined with Ready Systems component RTscope, a real-time multitasking debugger and VRTX32 system monitor.

Within the VRTX32/T model, extensions are provided which allow VRTX32/T tasks on different transputers to communicate over channels. This capability offers an effective way of constructing distributed real-time systems based on VRTX32/T, transputers and transputer communication links. VRTX32/T sets a new standard of performance, features and reliability in off-the-shelf software components. Ready Systems used its experience with over 3500 customers to develop VRTX32/T.

Deterministic Performance

In any real-time system, tasks must respond consistently and quickly to external events. In order to build an application that reacts reliably every time, the software developer must know the performance capabilities of the hardware and software components used in the system. VRTX32/T is designed to give consistent response times no matter how many tasks, queues, or mailboxes are added to the system. In addition, VRTX32/T does not exact a performance penalty when used with small systems. VRTX32 services are invoked through a C interface. System calls exist for task management, memory allocation, inter-task synchronization and communication, real-time clock support, character I/O, interrupt handling and VRTX32 initialisation.

Distributed Environment

VRTX32/T fully exploits the transputer communication technology. Systems consisting of both 16-bit and 32-bit transputers can easily be constructed to match the application need. The distributed nature of VRTX32/T means that, even in demanding applications, resource conflict can be avoided by easily adding additional processing nodes. This multiprocessing capability also lends itself to fault tolerant systems.

Support

VRTX32 is supported by Ready Systems' worldwide field and factory support organisation. All Ready Systems' products include a 90-day Standard Product Support service warranty. Upon expiration, customers may choose from a variety of support services including standard product technical phone support, on-line bulletin board, update service or consulting and product training.

INMOS products are supported worldwide through SGS-THOMSON Sales Offices, Regional Technology Centres, and authorised distributors.



C Executive

Advance Information

C Executive for IBM PC, NEC PC, Sun 3, Sun 4 and VAX hosts

DESCRIPTION

The C EXECUTIVE for the transputer allows you to design and implement real-time embedded systems based on single or multiple transputers.

KEY FEATURES

- Real-time, prioritised fully pre-emptive task scheduler.
- UNIX-style standard I/O and redirection of I/O.
- Optional DOS filesystem available with disk device drivers.
- Real-time clock support.
- General-purpose event mechanism.
- Allows multi-transputer communications via a virtual link driver.
- Includes the fully ROMable JMI Portable C Library.
- XON/XOFF support built-in.
- Interrupt-driven device drivers.
- Full duplex serial device drivers.
- Built-in support for configurable devices.
- Device driver for physical and virtual links.
- Device driver for console and disk I/O via INMOS iserver.
- Device driver for SCSI disk.
- All system calls available as direct C function calls.

16.1 C EXECUTIVE For The Transputer

C EXECUTIVE for the transputer has been developed to facilitate the design and implementation of transputer-based real-time embedded systems. It provides the transputer with a general-purpose interrupt mechanism; any channel (in-memory, physical link, virtual link) can be designated as a source of interrupt. Interrupt handlers can be assigned to interrupt sources. In addition, a periodic clock interrupt may be defined.

The same system calls are used for disk I/O, terminal I/O and interprocess communications, even where the communication is inter-processor. The I/O devices used by an application task may be reassigned by reconfiguration: no change to application code is required. This greatly simplifies system design and implementation; for example, a terminal may be used to simulate a process when testing the system. Since interprocess communication, inter-processor I/O and physical I/O are carried out using the same system calls, distributed systems can be very easily designed and implemented. In addition single processor systems can be easily scaled into multiprocessor systems.

C EXECUTIVE provides such services as fully pre-emptive scheduling, standard I/O, I/O redirection, terminal, disk, clock and link (physical and virtual) drivers, interprocess communication and, optionally, access to a filesystem. These services and a comprehensive portable C library are provided through re-entrant ROMable build-time libraries and startup code.

The user defined tasks are linked with the startup code and libraries to form an executable program. Apart from the scheduler and interrupt support code, other services are only extracted from the library if required. This means the program is only as large as it needs to be.

Those parts of the C EXECUTIVE which are specific to a particular processor are the interrupt handling, context switching, scheduling and device drivers. On the transputer, the C EXECUTIVE with user tasks and device drivers runs as a single level 1 process. Context switching and initial interrupt handling are carried out by level 0 processes.

C EXECUTIVE is a well established industrial strength real-time kernel which has been implemented for more than fifteen different target processor families. C EXECUTIVE for the transputer is initially packaged for the T2, T4 and T8 series transputers. A device driver is provided which supports both physical links and virtual links. The package has been designed to facilitate migration to the forthcoming series of transputers codenamed H1 at the time of writing.

16.2 Benefits Of C Executive

C EXECUTIVE is written almost entirely in ANSI Standard C - resulting in maximum reliability, portability and ease of maintenance. This means of implementation also provides the most efficient mechanism, both in terms of space and speed, for making system calls from high-level languages; extra layers of interface libraries are not required. Critical sections of C EXECUTIVE, including those responsible for context switching, task scheduling and interrupt handling, have been implemented in assembly language. This mixture of C and assembly languages provides both a C programming environment and fast response times for interrupts and context switching.

Multiprocessor systems can be configured for the transputer - application tasks executing on separate processors can communicate with each other via virtual link drivers. The standard development package allows an application to be built for either a T2, T4 or T8 series transputer.

C EXECUTIVE supports an optional filesystem - CE-DOSFILE is designed for optimum performance in real-time data acquisition applications. Major extensions to the standard DOS filesystem give improved performance and include contiguous files and configurable caches for each disk device.

CE-DOSFILE maintains complete media compatibility with DOS - using C EXECUTIVE and CE-DOSFILE on any CPU architecture, removable media can be read directly by any DOS system. CE-DOSFILE automatically maintains 8086 byte order within all filesystem structures, regardless of the host CPU byte ordering.

C EXECUTIVE can be extensively customised - it may be regarded as a very flexible component of the target system.

C EXECUTIVE encourages and facilitates dataflow design methods – viewing a system as a set of independent asynchronous processes, communicating with each other and with the external environment via clearly defined input and output data streams. The data queue mechanism provided by C EXECUTIVE allows direct implementation of dataflow design. Data queues are not limited by message size or number of messages, as with traditional “mailboxes”. Messages of various sizes can pass through the same data queue, with end-of-message events automatically alerting waiting tasks.

C EXECUTIVE provides built-in support for multiple terminals – all standard terminal input/output facilities, including XON/XOFF are simple configuration options, requiring no user-written code. Any combination of full duplex and half duplex devices can be configured.

Resource co-ordination – provided by a general-purpose named resource locking/unlocking mechanism. A shared text facility helps conserve memory by allowing each task to be represented by one or more processes, each with a separate priority. Shared text saves system memory, whilst retaining a private data area for each process. Up to 32768 discrete process priorities may be assigned.

C EXECUTIVE includes a comprehensive portable C library – contains almost 100 routines for memory allocation, character string manipulation, and buffered and formatted I/O. The JMI Portable C Library is the equivalent of a normal UNIX portable library, but is ROMable, sharable and re-entrant.

No special software development tools required – C EXECUTIVE is compiled, assembled and linked using the same tools as are used for the application programs.

16.3 CE-VIEW

CE-VIEW is an interactive system level debugging tool, which enables the user to “view” and modify the overall target system configuration, and also control individual user tasks within the system.

16.4 Documentation

C EXECUTIVE packages are supplied with an installation, release notes and comprehensive user documentation.

16.5 Availability

The binary development copy of C EXECUTIVE for the transputer is packaged for the INMOS C compiler host on PC (IMS D7214), VAX/VMS (IMS D6214), Sun 3 (IMS D5214) or Sun 4 (IMS D4214). The package supports T2, T4 and T8 series transputers.

Three software packages are available:

- Basic C EXECUTIVE for the transputer Optional CE-DOSFILE Optional CE-VIEW

The packages are provided on media appropriate to the host system (PC, VAX/VMS, Sun 3 or Sun 4).

C EXECUTIVE for the transputer provides the following device drivers:

iserver tty driver link driver supporting both physical and virtual links

The optional CE-DOSFILE package provides the following additional device driver:

- disk driver for the INMOS B422 SCSI TRAM memory resident disk driver iserver disk driver

The C EXECUTIVE for the transputer is expected to be available before the end of 1990.

16.6 Licensing

The development copy of the transputer C EXECUTIVE is supplied under a shrinkwrap licence agreement. This licence allows the development copy to be installed on a single host development system, targeting

a system incorporating copies of C EXECUTIVE on up to two transputers. Development copies are available from Real Time Systems and its distributors in USA, France and Japan. Licences for multi-transputer systems, source, OEM and unlimited use are also available.

16.7 How To Order

Please contact the Real Time Systems Limited or the distributor appropriate to your area: UK: Real Time Systems Limited, PO Box 70, Viking House, Nelson Street, Douglas, Isle of Man, tel: (+44) 624 661400, fax: (+44) 624 663453.

France: COSMIC, 33 rue Le Corbusier, Europarc Creteil, 94035 Creteil, Cedex, tel: (+33) 1 43 99 53 90, fax: (+33) 1 43 99 14 83.

Japan: Advance Data Controls Corp., Nihon Seimei Otsuka Bldg., No. 13-4, Kita Otsuka 1-chome, Toshima-ku, Tokyo, tel (+81) 3 576 5351, fax (+81) 3 576 1772.

USA: JMI Software Consultants, Inc., PO Box 481, 904 Sheble Lane, Spring House PA 19477, tel: (+1) 215 628 0840, fax: (+1) 215 628 0353.

NOTE: C EXECUTIVE is a registered trademark of JMI Software Consultants, Inc. in the USA. CE-DOSFILE and CE-VIEW are trademarks of JMI Software Consultants, Inc.

© Copyright Real Time Systems Limited 1990

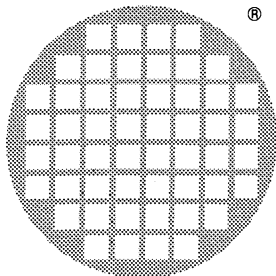
All information on C Executive is provided by Real Time Systems Limited. Although every effort has been made, Inmos does not guarantee the accuracy of the contents of this section.



Hardware Products



TRAnsputer Modules (TRAMS)

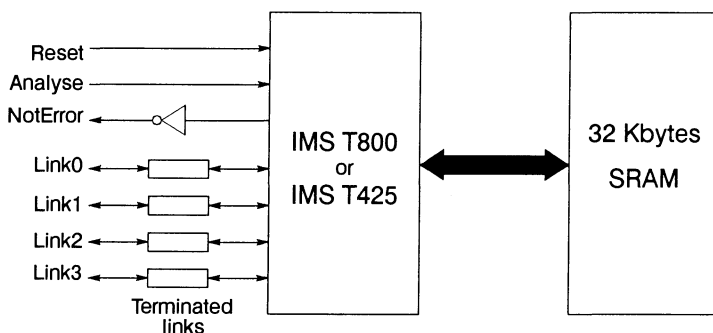


inmos®

IMS B401 TRAM

32-bit transputer
32 Kbytes
Size 1

Engineering Data



FEATURES

- Choice of Transputer (IMS T425 or IMS T800)
- Choice of Processor Speed (20 or 25 MHz)
- 32 Kbytes of no-wait-state SRAM
- Communicates via 4 INMOS serial links (Selectable between 10 or 20 Mbits/s)
- Package has only 16 active pins.
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

A low cost, high performance, 16 pin transputer, ideal for applications where 4 Kbytes of on-chip RAM is not quite enough. The 32 Kbytes of off chip RAM is ideal for systolic processing, signal processing, feature extraction etc. The IMS B008, fitted with ten IMS B401-5s, offers 50 MWhetstones/s in a single slot of an IBM PC, XT, AT, PS2 model 30, or clone.

17.1 IMS B401 TRAM engineering data

17.1.1 Introduction

The IMS B401 is one of a range of INMOS TRAnsputer Modules (TRAMs) incorporating a transputer and 32 Kbytes of static RAM. In effect, these TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

17.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7

Table 17.1 IMS B401 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Figure 17.3.

17.1.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached, but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. (The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

LinkspeedA and LinkspeedB (pins 6 and 7)

LinkspeedA and **LinkspeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low the link(s) operate at 10 Mbits/s and when high the link(s) operate at 20 Mbits/s.

Link signals

Whilst the links obey a protocol identical to that described in *The Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

17.1.4 Memory configuration

The IMS B401 with a IMS T414 has internal RAM occupying the first 2 Kbytes of address space, whereas internal RAM occupies the first 4 Kbytes on the IMS B401 with an IMS 425 or IMS T800. The two memory maps shown in Figure 17.1 reflect this fact.

Location of external memory

The IMS B401 has 32 Kbytes of external memory. Tables 17.2 and 17.3 show the start addresses of this memory for both the IMS T414 and the IMS T425/T800 versions of the IMS B401 (the “#” sign indicates a hexadecimal number).

	Hardware byte address
From:	#80000800
To:	#800087FF

Table 17.2 Location of external memory on the IMS B401 with IMS T414

	Hardware byte address
From:	#80001000
To:	#80008FFF

Table 17.3 Location of external memory on the IMS B401 with IMS T425 or IMS T800

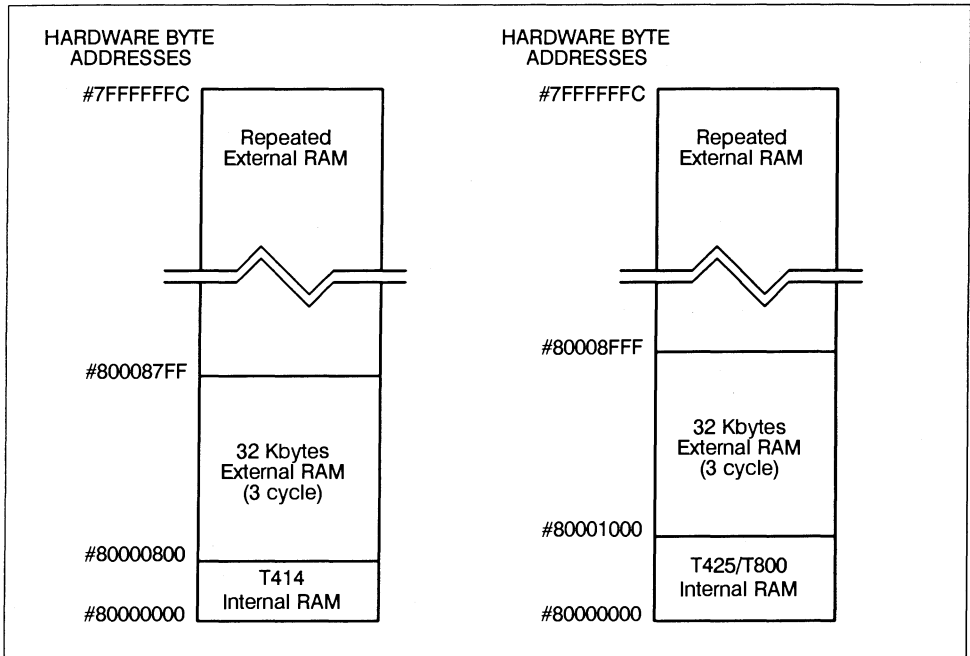


Figure 17.1 Memory maps

17.1.5 Mechanical details

Figure 17.2 indicates the vertical dimensions of an IMS B401 TRAM and Figure 17.3 shows the outline drawing of the IMS B401.

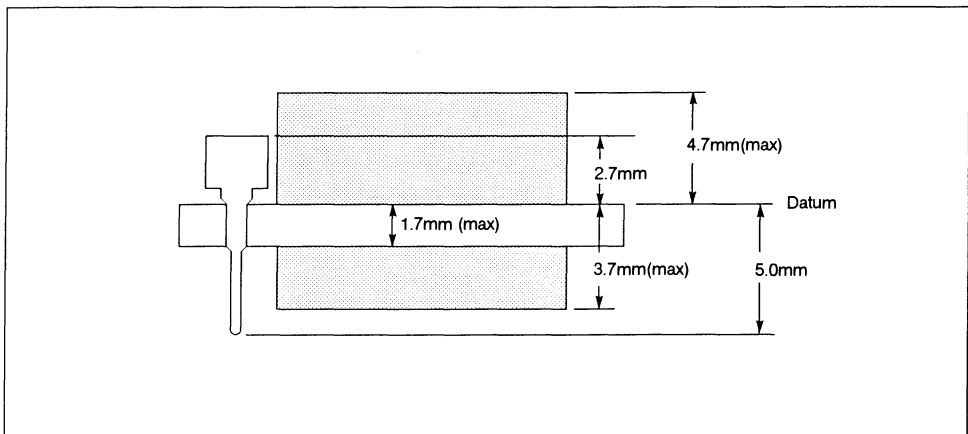


Figure 17.2 IMS B401 height specification

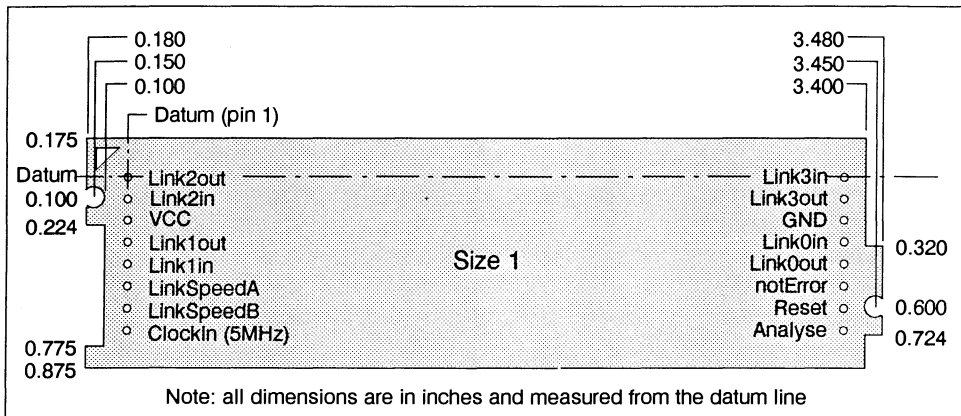


Figure 17.3 IMS B401 outline drawing

17.1.6 Installation

Since the IMS B401 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B401 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B401 carefully into the motherboard. Where the IMS B401 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B401 (see Figure 17.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot. If it is envisaged that the assembly is likely to be subjected to any vibrations, it is recommended that the TRAM is secured to the motherboard using nylon M3 nuts and bolts. The bolts should be inserted through the fixing holes on the motherboard, and through the castellations on two edges of the TRAM. A number of these nuts and bolts are supplied with each of the INMOS motherboards.

Should it be necessary to unplug the IMS B401, it is advised that, having removed any retaining nuts and bolts, it is gently levered out while keeping it as flat as possible. As soon as the IMS B401 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

17.1.7 Specification

TRAM feature	IMS B401-3	IMS B401-8	IMS B401-5	Unit	Notes
Transputer type	T800-20	T425-25	T800-25		
Number of transputers	1	1	1		
Number of INMOS serial links	4	4	4		
Amount of SRAM	32	32	32	Kbytes	
SRAM "wait states"	0	0	0		
SRAM cycle time	150	120	120	ns	
Amount of DRAM	None	None	None		
DRAM "wait states"	N/A	N/A	N/A		
DRAM cycle time	N/A	N/A	N/A		
Subsystem controller	No	No	No		
Peripheral circuitry	None	None	None		
Parity	No	No	No		
Size (TRAM size)	1	1	1		
Length	3.66	3.66	3.66	inch	
Pitch between pins	3.30	3.30	3.30	inch	
Width	1.05	1.05	1.05	inch	
Component height above PCB	4.7	4.7	4.7	mm	
Component height below PCB	3.7	3.7	3.7	mm	1
Weight	21	21	21	g	
Storage temperature	0-70	0-70	0-70	°C	
Operating temperature	0-50	0-50	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	4.75-5.25	4.75-5.25	Volt	
Power consumption	1.2	1.5	1.5	W	3

Table 17.4 IMS B401 specification

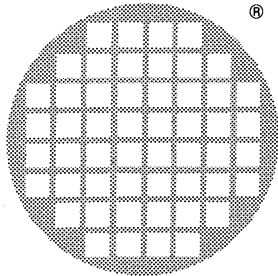
Notes:

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B401 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

17.1.8 Ordering Information

Description	Order Number
IMS B401 TRAM with IMS T800-20	IMS B401-3
IMS B401 TRAM with IMS T425-25	IMS B401-8
IMS B401 TRAM with IMS T800-25	IMS B401-5

Table 17.5 Ordering information

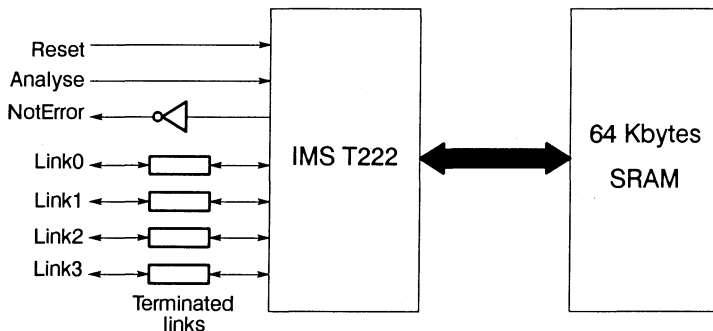


inmos[®]

IMS B416 TRAM

16-bit transputer
64 Kbytes
Size 1

Engineering Data



FEATURES

- IMS T222 Transputer
- 64 Kbytes of no-wait-state SRAM (100 ns memory cycle time)
- Communicates via 4 INMOS serial links (Selectable between 10 or 20 Mbits/s)
- Package has only 16 active pins.
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The IMS B416 utilises the full memory space of the IMS T222 transputer. It is manufactured fully from surface mount silicon components. The IMS T222's PLCC package brings low cost benefits to TRAM users.

18.1 IMS B416 TRAM engineering data

18.1.1 Introduction

The IMS B416 is one of a range of INMOS TRANputer Modules (TRAMs) incorporating a transputer and 64 Kbytes of static RAM. In effect, these TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then the *Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

18.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC,GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7

Table 18.1 IMS B416 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Figure 18.3.

18.1.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached, but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in the *Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. (The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

LinkspeedA and LinkspeedB (pins 6 and 7)

LinkspeedA and **LinkspeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low, the link(s) operate at 10 Mbits/s and when high the links operate at 20 Mbits/s.

Link signals

Whilst the links obey a protocol identical to that described in the *Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

18.1.4 Memory configuration

The IMS B416 has internal RAM occupying the first 4 Kbytes of address space. The next 60 Kbytes is occupied by the external static RAM present on the board. The total of 64 Kbytes represents the maximum addressable memory space of the IMS T222 transputer.

Table 18.2 details the start and end addresses of this external memory and Figure 18.1 shows a graphical representation of the memory map (the “#” sign indicates a hexadecimal number).

	Hardware byte address
From:	#9000
To:	#7FFE

Table 18.2 Location of external memory on the IMS B416

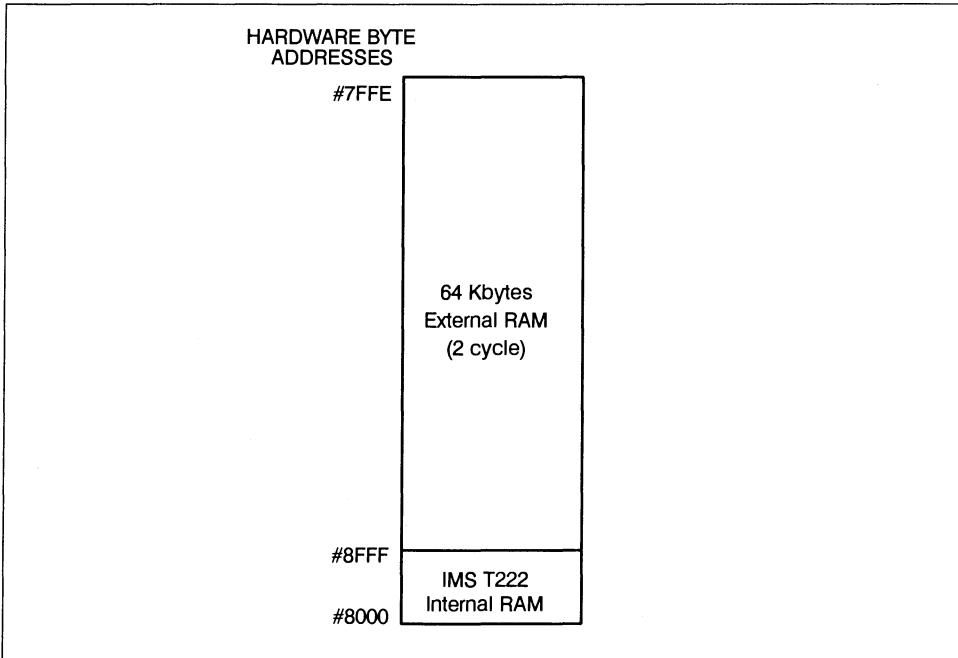


Figure 18.1 Memory map

18.1.5 Mechanical details

Figure 18.2 indicates the vertical dimensions of an IMS B416 TRAM and Figure 18.3 shows the outline drawing of the IMS B416.

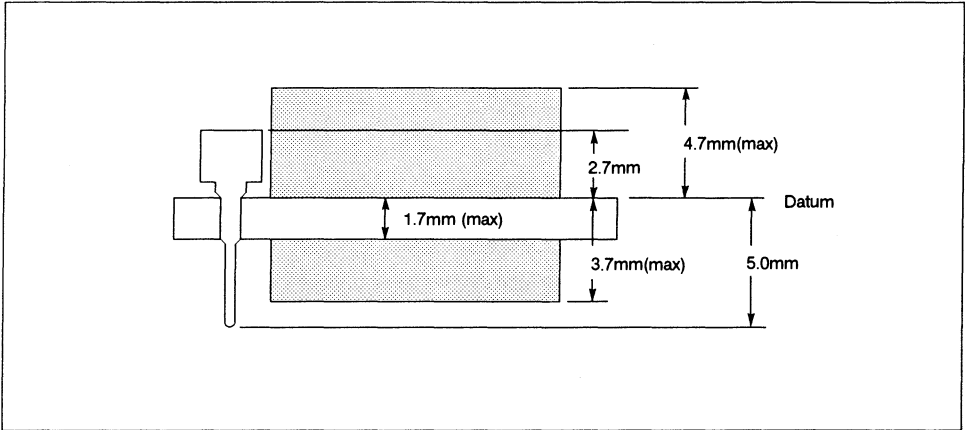
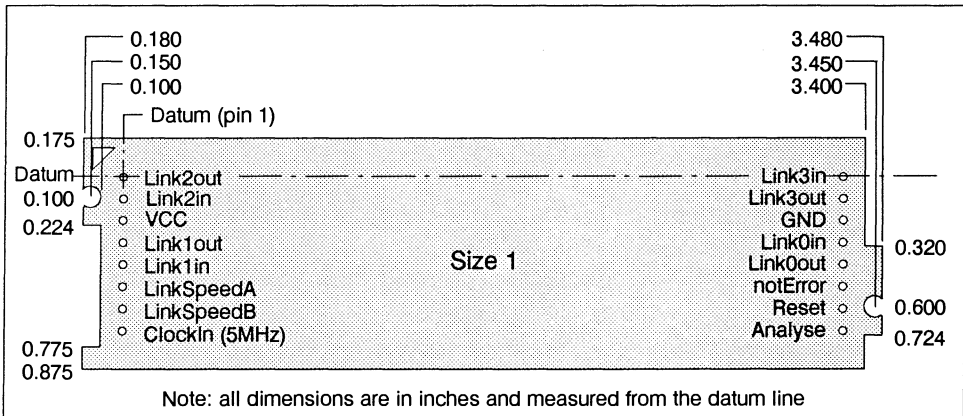


Figure 18.2 IMS B416 height specification



Note: all dimensions are in inches and measured from the datum line

Figure 18.3 IMS B416 outline drawing

18.1.6 Installation

Since the IMS B416 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B416 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B416 carefully into the motherboard. Where the IMS B416 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B416 (see Figure 18.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B416, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B416 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

18.1.7 Specification

TRAM feature	IMS B416-10	Unit	Notes
Transputer type	IMS T222-20		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	64	Kbytes	
SRAM "wait states"	0		
Amount of DRAM	None		
DRAM "wait states"	N/A		
Memory cycle time	100	ns	
Subsystem controller	No		
Peripheral circuitry	None		
Parity	No		
Size (TRAM size)	1		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	1.05	inch	
Component height above PCB	4.7	mm	
Component height below PCB	3.7	mm	1
Weight	30	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	2	W	3

Figure 18.4 IMS B416 specification

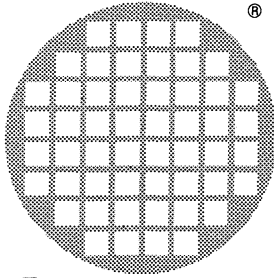
Notes:

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B416 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

18.1.8 Ordering Information

Description	Order No.
IMS B416 TRAM with IMS T222-20	IMS B416-10

Table 18.3 Ordering information



inmos[®]

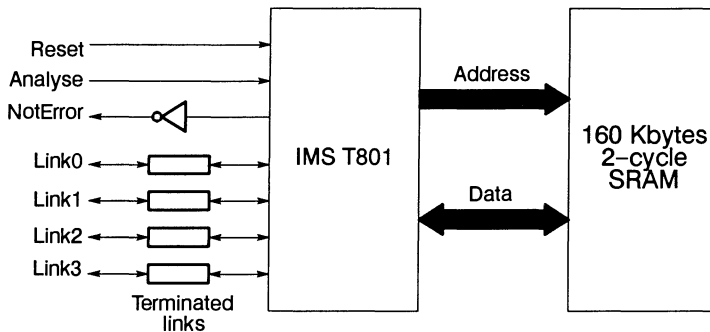
IMS B410 TRAM

32-bit transputer

160 Kbytes

Size 2

Engineering Data



FEATURES

- IMS T801 Transputer with demultiplexed address and data busses
- 160 Kbytes of no-wait-state SRAM (100 ns memory cycle time)
- Size 2 TRAM
- Communicates via 4 INMOS serial links
- Package has only 16 active pins.
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The de-multiplexed address and data busses of the IMS T801 transputer allow very high performance systems to be constructed. The IMS B410 TRAM achieves 2-cycle memory accesses with very fast SRAMs, and yet still manages to squeeze 160 Kbytes onto a size 2 TRAM.

The IMS B410 TRAM allows users to benchmark the performance of the IMS T801 transputer. The standard TRAM interface means that the processor can simply be plugged into existing development systems. However, this module is as equally at home in very high performance system products, as it is in the evaluation environment.

19.1 IMS B410 TRAM engineering data

19.1.1 Description

The IMS B410 is an INMOS Transputer Module (TRAM) incorporating the IMS T801 transputer and 160 Kbytes of static RAM. The demultiplexed address and data busses of the IMS T801 allow maximum use to be made of some very fast static RAM, and the IMS B410 achieves 2-cycle, 100ns external memory cycles.

TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort. TRAMs may be plugged into motherboards, which provide the necessary electrical signals, mechanical support and usually, an interface to a host machine. Various motherboards are now available from INMOS and from third-party vendors for most of the common computing platforms.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included in Part 3 of this databook. If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

19.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7

Table 19.1 IMS B410 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Figure 19.3.

19.1.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached, but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

LinkspeedA and LinkspeedB (pins 6 and 7)

LinkspeedA and **LinkspeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low the link(s) operate at 10 Mbits/s and when high the link(s) operate at 20 Mbits/s.

Link signals

Whilst the links obey a protocol identical to that described in *The Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

19.1.4 Memory configuration

The internal RAM of the IMS T801 occupies the first 4 Kbytes of address space. The next 156 Kbytes is occupied by the external static RAM present on the TRAM. There is then a gap of 96 Kbytes. A pattern of 160 K of external RAM followed by a gap is repeated in 256 Kbyte blocks throughout the higher address space.

Table 19.2 details the start and end addresses of the external memory and Figure 19.1 shows a graphical representation of the memory map (the “#” sign indicates a hexadecimal number).

	Hardware byte address
From:	#80001000
To:	#80027FFF

Table 19.2 Location of external memory on the IMS B410

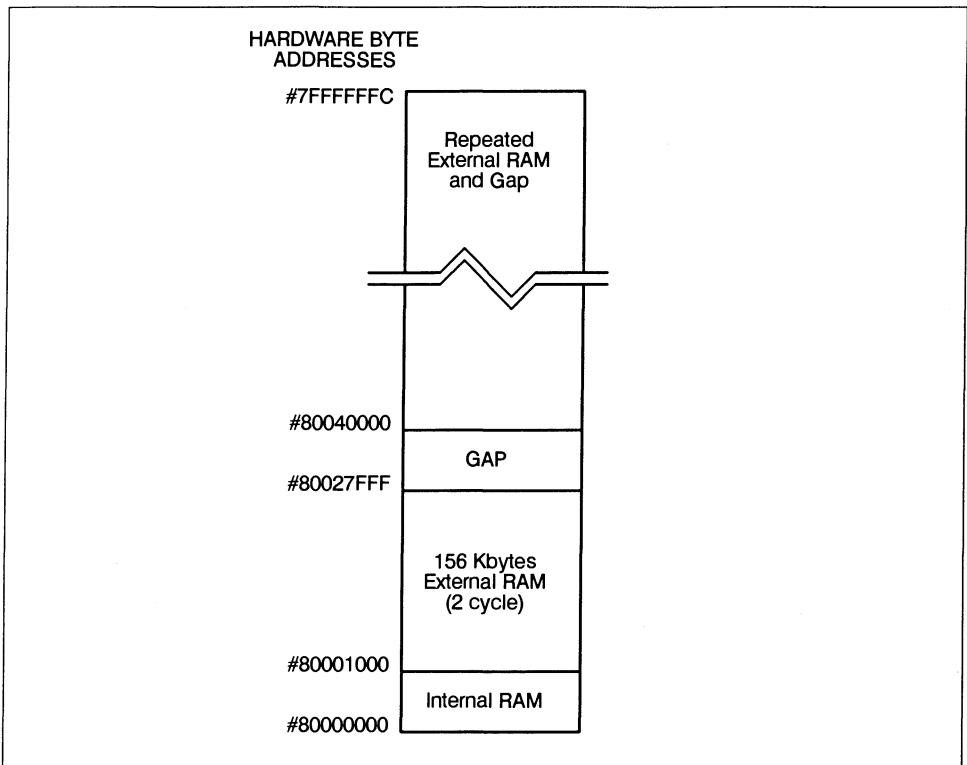


Figure 19.1 Memory map

19.1.5 Mechanical details

Figure 19.2 indicates the vertical dimensions of an IMS B410 TRAM and Figure 19.3 shows the outline drawing of the IMS B410.

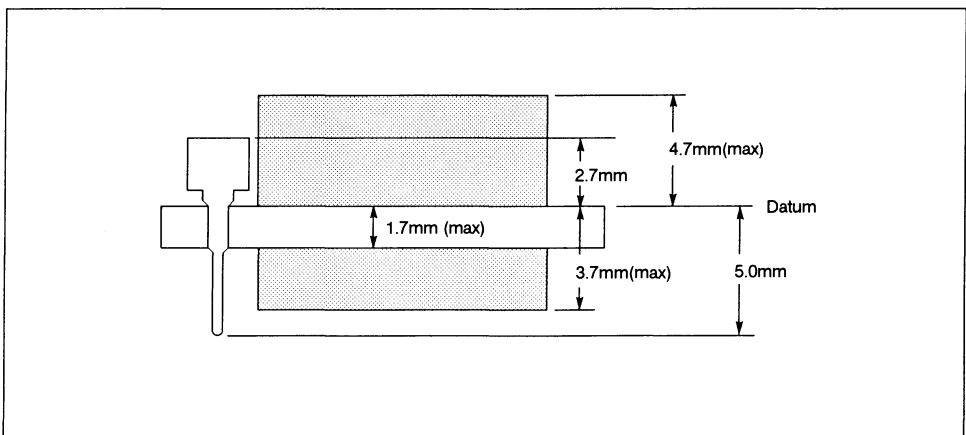


Figure 19.2 IMS B410 height specification

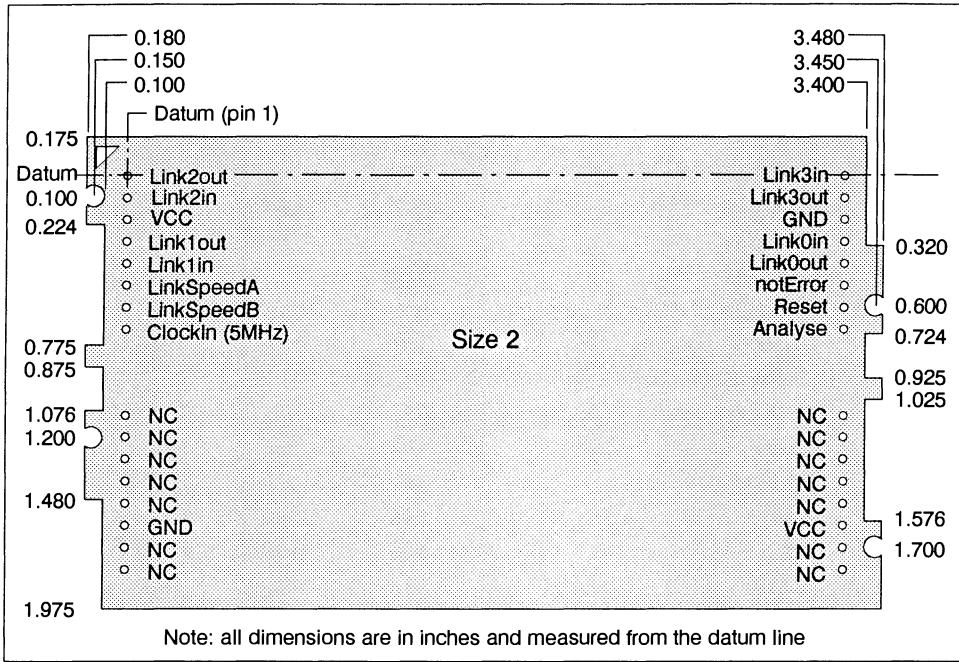


Figure 19.3 IMS B410 outline drawing

19.1.6 Installation

Since the IMS B410 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B410 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B410 carefully into the motherboard. Where the IMS B410 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B410 (see Figure 19.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot. If it is envisaged that the assembly is likely to be subjected to any vibrations, it is recommended that the TRAM is secured to the motherboard using nylon M3 nuts and bolts. The bolts should be inserted through the fixing holes on the motherboard, and through the castlations on two edges of the TRAM. A number of these nuts and bolts are supplied with each of the INMOS motherboards.

Should it be necessary to unplug the IMS B410, it is advised that, having removed any retaining nuts and bolts, it is gently levered out while keeping it as flat as possible. As soon as the IMS B410 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

19.1.7 Specification

TRAM feature	IMS B410-11	Unit	Notes
Transputer type	T801-20		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	160	Kbytes	
SRAM "wait states"	0		
Amount of DRAM	None		
DRAM "wait states"	N/A		
Memory cycle time	100	ns	
Subsystem controller	No		
Peripheral circuitry	None		
Parity	No		
Size (TRAM size)	2		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	2.15	inch	
Component height above PCB	4.7	mm	
Component height below PCB	3.7	mm	1
Weight	50	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	3	W	3

Table 19.3 IMS B410 specification

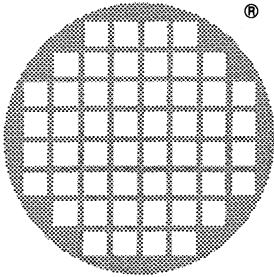
Notes:

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value utilising all four links and with memory accessed simultaneously at a supply voltage (VCC) of 5.25 V.

19.1.8 Ordering Information

Description	Order Number
IMS B410 TRAM with IMS T801-20	IMS B410-11

Table 19.4 Ordering information

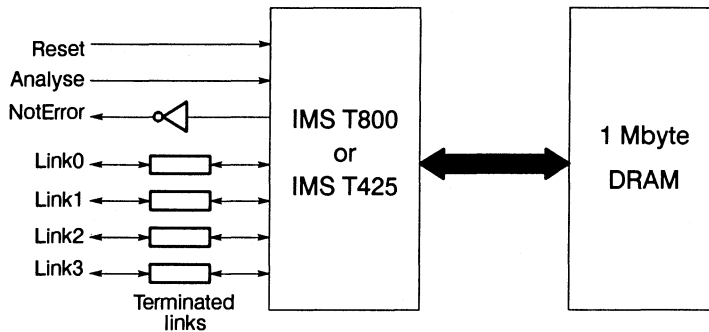


inmos®

IMS B411 TRAM

32-bit transputer
1 Mbyte
Size 1

Engineering Data



FEATURES

- Choice of IMS T425 or IMS T800 Transputer
- 1 Mbyte of zero wait-state DRAM (150 ns memory cycle time)
- Size 1 TRAM
- Communicates via 4 INMOS serial links
- Package has only 16 active pins.
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The IMS B411 TRAM is the ideal module for applications where space is at a premium. With a full Mbyte of DRAM on a size 1 TRAM, 8 transputers and 8 Mbytes of memory can be installed in a single IBM PC (*) slot (using the IMS B008 motherboard) or in a single 6U VMEbus slot (using the IMS B014 motherboard). The choice of an IMS T800 or T425 transputer gives the user the flexibility to tailor a system to his exact requirements in terms of cost and performance.

(*) For IBM PC read: original PC, XT, AT, PS/2 Model 30 and most clones.

20.1 IMS B411 TRAM engineering data

20.1.1 Description

The IMS B411 is an INMOS TRANputer Module (TRAM) incorporating either an IMS T800 or IMS T425 transputer and 1 Mbyte of dynamic RAM.

TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort. TRAMs may be plugged into motherboards, which provide the necessary electrical signals, mechanical support and usually, an interface to a host machine. Various motherboards are now available from INMOS and from third-party vendors for most of the common computing platforms.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

20.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7

Figure 20.1 IMS B411 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Figure 20.4.

20.1.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached, but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

LinkSpeedA and LinkSpeedB (pins 6 and 7)

LinkSpeedA and **LinkSpeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low the link(s) operate at 10 Mbits/s and when high the link(s) operate at 20 Mbits/s.

Link signals

Whilst the links obey a protocol identical to that described in the *Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

20.1.4 Memory configuration

The internal RAM of the IMS T800 or IMS T425 occupies the first 4 Kbytes of address space. The next 1 Mbyte is occupied by the external dynamic RAM present on the TRAM. The external RAM is repeated in 1 Mbyte blocks throughout the higher address space.

Table 20.1 details the start and end addresses of the external memory and Figure 20.2 shows a graphical representation of the memory map (the “#” sign indicates a hexadecimal number).

Hardware byte address	
From:	#80001000
To:	#80100FFF

Table 20.1 Location of external memory on the IMS B411

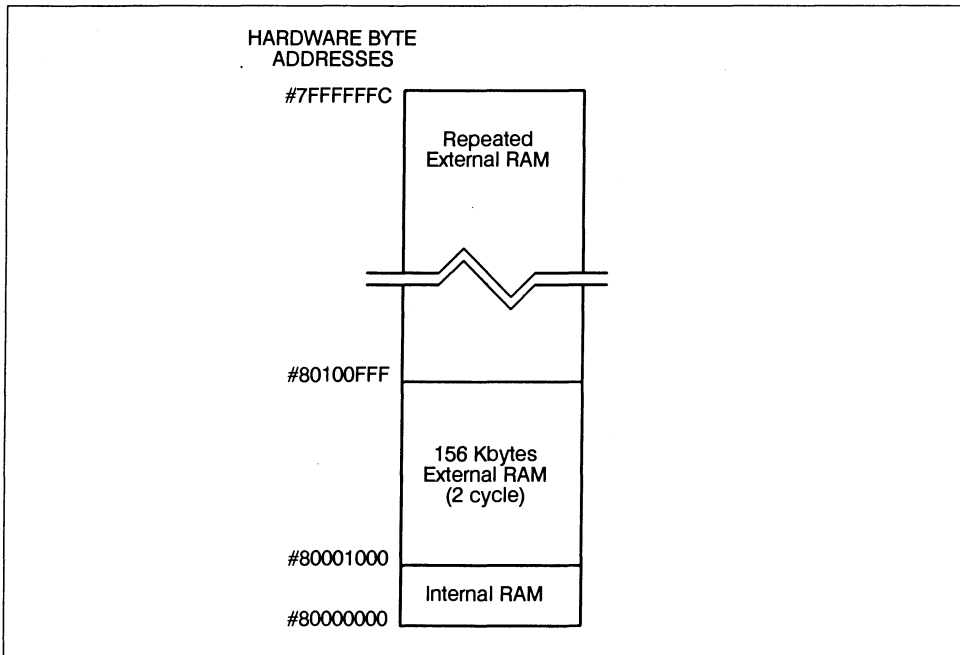


Figure 20.2 Memory map

20.1.5 Mechanical details

Figure 20.3 indicates the vertical dimensions of a single IMS B411 TRAM, and Figure 20.4 shows the outline drawing of the IMS B411.

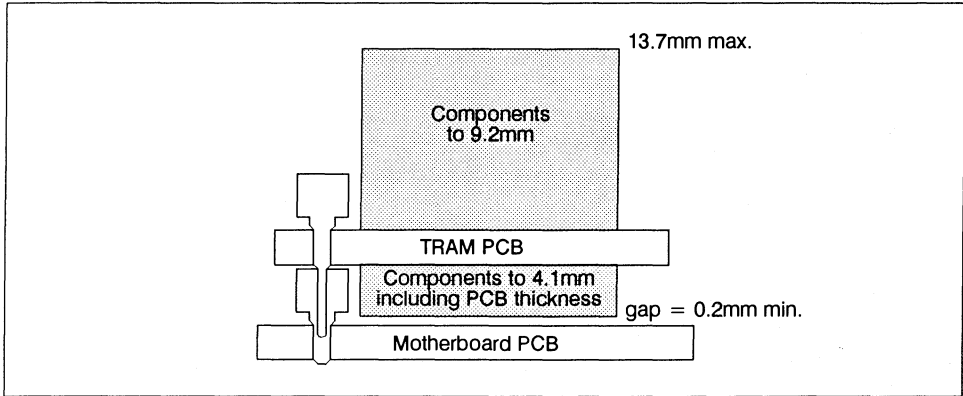


Figure 20.3 IMS B411 height specification

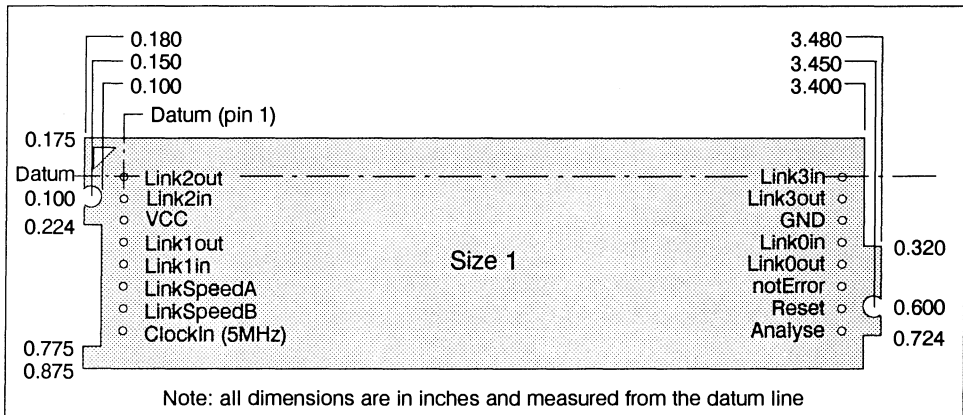


Figure 20.4 IMS B411 outline drawing

20.1.6 Installation

Since the IMS B411 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B411 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B411 carefully into the motherboard. Where the IMS B411 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B411 (see Figure 20.4) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot. If it is envisaged that the assembly is likely to be subjected to any vibrations, it is recommended that the TRAM is secured to the motherboard using nylon M3 nuts and bolts. The bolts should be inserted through the fixing holes on the motherboard, and through the castlignations on two edges of the TRAM. A number of these nuts and bolts are supplied with each of the INMOS motherboards.

Should it be necessary to unplug the IMS B411, it is advised that, having removed any retaining nuts and bolts, it is gently levered out while keeping it as flat as possible. As soon as the IMS B411 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

20.1.7 Specification

TRAM feature	IMS B411-3	IMS B411-7	Unit	Notes
Transputer type	T800-20	T425-20		
Number of transputers	1	1		
Number of INMOS serial links	4	4		
Amount of SRAM	None	None		
SRAM "wait states"	N/A	N/A		
Amount of DRAM	1	1	Mbyte	
DRAM "wait states"	0	0		
Memory cycle time	150	150	ns	
Subsystem controller	No	No		
Peripheral circuitry	None	None		
Parity	No	No		
Size (TRAM size)	1	1		
Length	3.66	3.66	inch	
Pitch between pins	3.30	3.30	inch	
Width	1.05	1.05	inch	
Component height above PCB	9.2	9.2	mm	
Component height below PCB	3.7	3.7	mm	1
Weight	50	50	g	
Storage temperature	0-70	0-70	°C	
Operating temperature	0-50	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	4.75-5.25	Volt	
Power consumption	4	4	W	3

Table 20.2 IMS B411 specification

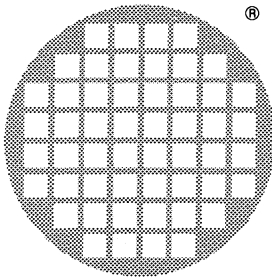
Notes:

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B411 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

20.1.8 Ordering Information

Description	Order Number
IMS B411 TRAM with IMS T800-20	IMS B411-3
IMS B411 TRAM with IMS T425-20	IMS B411-7
IMS B411 TRAM with IMS T800-25	IMS B411-5
IMS B411 TRAM with IMS T425-25	IMS B411-8

Table 20.3 Ordering information



inmos[®]

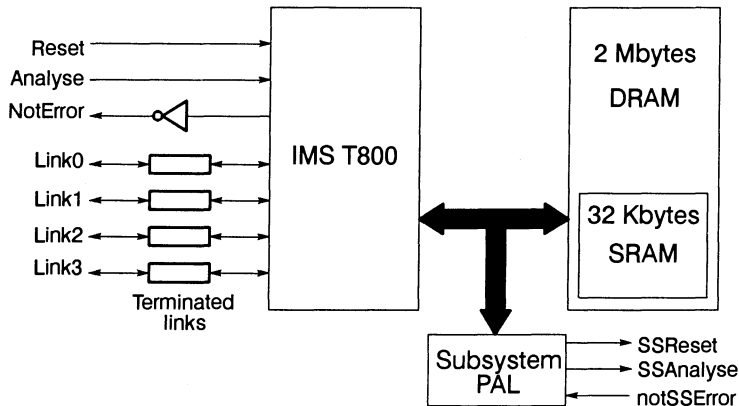
IMS B404 TRAM

32-bit transputer

2 Mbytes

Size 2

Engineering Data



FEATURES

- IMS T800 Transputer
- 32 Kbytes of zero wait-state SRAM
- 2 Mbytes of single wait-state DRAM
- Subsystem controller circuitry
- Communicates via 4 INMOS serial links (Selectable between 10 or 20 Mbits/s)
- Package has only 16 active pins
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The IMS B404 is a very compact compute module providing a full 2 Mbytes of memory and still providing maximum performance capability. This is achieved by extending the principle of fast on chip RAM to include 32 Kbytes of static RAM which cycles as fast as possible. So any technique which puts most frequently accessed memory locations near the bottom of memory will speed up the processing. This TRAM is the most popular board for running INMOS' Toolset packages.

21.1 IMS B404 TRAM engineering data

21.1.1 Introduction

The IMS B404 is one of a range of INMOS TRANputer Modules (TRAMs) incorporating a IMS T800 transputer, 32 Kbytes of static RAM and 2 Mbytes of dynamic RAM. In effect, these TRAMs are board level transputers with a simple, standardized interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

21.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7
Subsystem Services			
SubSystemReset	out	Subsystem reset	1b
SubSystemAnalyse	out	Subsystem error analysis	1c
notSubSystemError	in	Subsystem error indicator	1a

Table 21.1 IMS B404 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Figure 21.3.

21.1.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached, but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. (The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

LinkSpeedA and LinkSpeedB (pins 6 and 7)

LinkSpeedA and **LinkSpeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low, the link(s) operate at 10 Mbits/s, and when high the link(s) operate at 20 Mbits/s.

Link signals

Whilst the links obey a protocol identical to that described in the *Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

21.1.4 Subsystem signals

The IMS B404 has a subsystem port in addition to the usual TRAM signals. This enables the TRAM to reset or analyse a subsystem of other TRAMs and/or motherboards. The polarity of these signals is the same as that of the **Reset**, **Analyse** and **notError** standard TRAM signals. Therefore, the IMS B404 subsystem can drive other TRAMs on the same motherboard with no intermediate logic. However, **SubSystemReset** and **SubSystemAnalyse** must go through inverting buffers if they are to drive a subsystem off the motherboard.

These subsystem signals are accessed by writing or reading to control registers in the transputer memory space.

21.1.5 Memory configuration

The IMS B404 is able to access 2 Mbytes of memory. This is comprised of 4 Kbytes of internal transputer memory, 28 Kbytes of external SRAM and 2016 Kbytes of external DRAM. There are, in fact, 32 Kbytes of SRAM components and 2 Mbytes of DRAM components on the board, but the address spaces of each type of memory are superimposed. Therefore, the total memory available is limited to 2 Mbytes. This is sufficient to enable the Transputer Development System (TDS) to be run on a single IMS B404 TRAM.

Location of external memory

Tables 21.2 and 21.3 show the start addresses of the different types of external memory on the IMS B404 (the “#” sign indicates a hexadecimal number). The internal RAM on the IMS T800 occupies the first 4 Kbytes of address space.

	Hardware byte address
From:	#80001000
To:	#80007FFF

Table 21.2 Location of external SRAM on the IMS B404

	Hardware byte address
From:	#80008000
To:	#801FFFFF

Table 21.3 Location of external DRAM on the IMS B404

Since the internal memory on the IMS T800 is 1 cycle, the external SRAM is 3 cycle and the DRAM is 4 (or 5) cycle, a memory speed hierarchy is established. This architecture allows programmers to structure their code for optimum performance, and will become of even greater significance when the next faster version of the transputer becomes available.

Subsystem register locations

The subsystem register addresses start at hardware address #00000000 in all TRAMs that utilize a 32-bit processor, allowing software compatibility between TRAMs. These registers are located as shown in Table 21.4.

Register	Hardware byte address
SubSystemReset (write only)	#00000000
SubSystemAnalyse (write only)	#00000004
notSubSystemError (read only)	#00000000

Table 21.4 Subsystem address locations

Setting bit 0 in either the reset or the analyse registers asserts the corresponding signal. Similarly, clearing bit 0 deasserts the signal. When an error occurs in the subsystem, bit 0 of the error location becomes set.

Byte locations #00000008 and #0000000C are unused. The subsystem registers are repeated at every sixteenth byte location in the positive address space. See Figure 21.1.

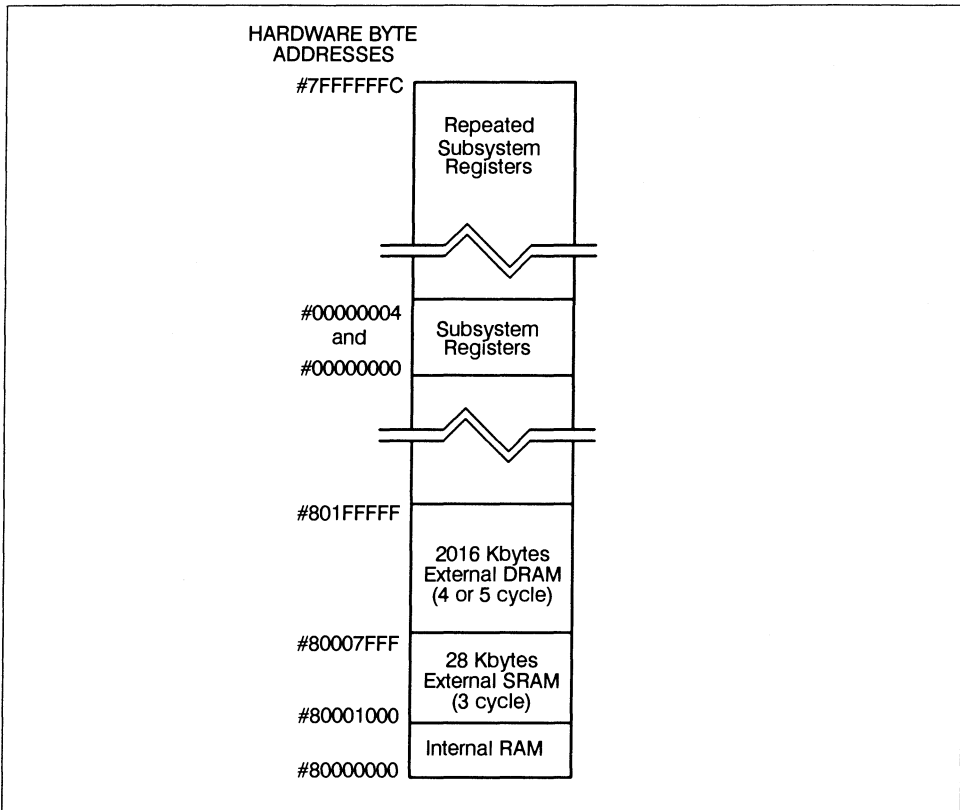


Figure 21.1 Memory map

21.1.6 Mechanical details

Figure 21.2 indicates the vertical dimensions of a single IMS B404 and Figure 21.3 shows the outline drawing the of the IMS B404.

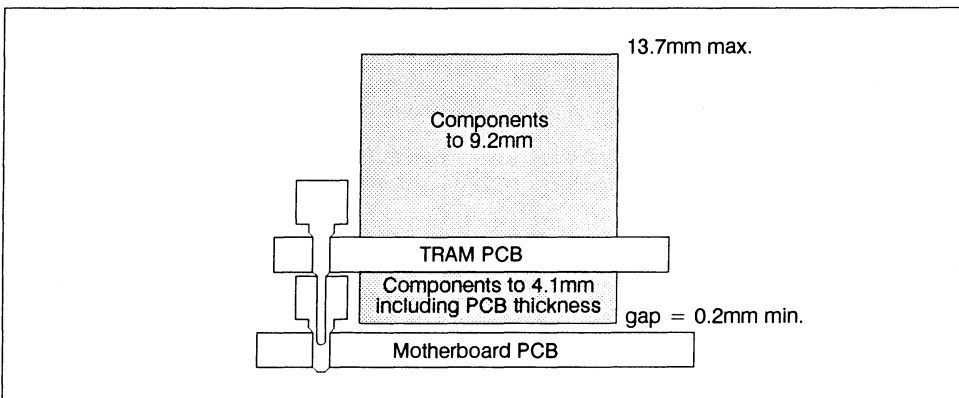


Figure 21.2 IMS B404 height specification

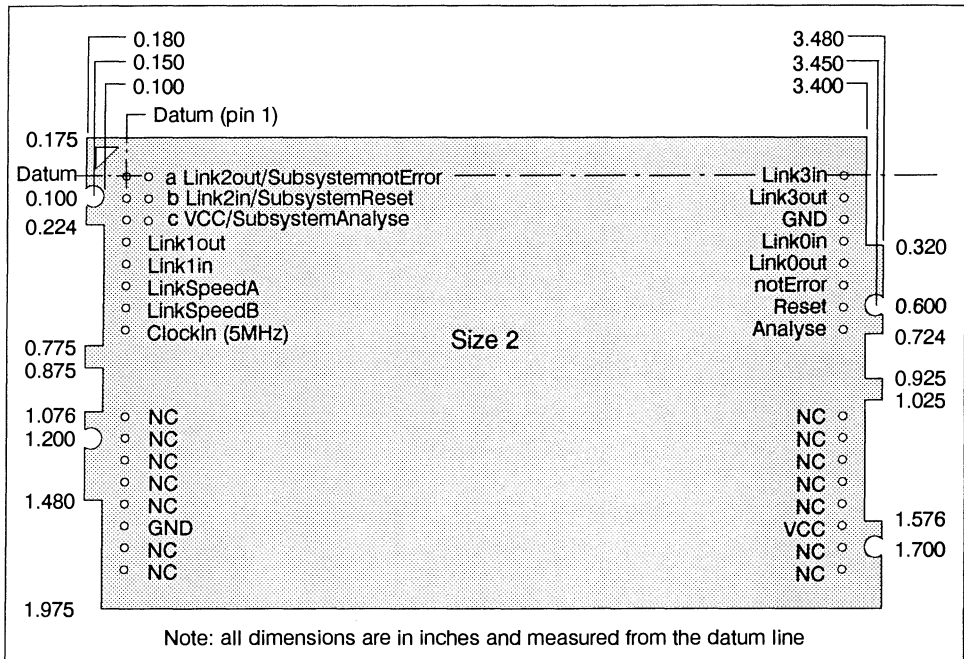


Figure 21.3 IMS B404 outline drawing

21.1.7 Installation

Since the IMS B404 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B404 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

If the subsystem signals are required, plug a 3-way header strip into the solder-side sockets on the IMS B404.

Plug the IMS B404 into the motherboard. Where the IMS B404 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B404 (see Figure 21.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B404, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B404 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

21.1.8 Specification

This specification applies to part numbers IMS B404-5x, IMS B404-3y and IMS B404-8z, where *x* denotes Rev. A and after, *y* denotes Rev. E and after, and *z* denotes Rev. C and after.

TRAM feature	IMS B404-5	IMS B404-3	IMS B404-8	Unit	Notes
Transputer type	IMS T800-25	IMS T800-20	IMS T425-25		
Number of transputers	1	1	1		
Number of INMOS serial links	4	4	4		
Amount of SRAM	32	32	32	Kbyte	
SRAM "wait states"	0	0	0		
Amount of DRAM	2	2	2	Mbyte	
DRAM "wait states"	1(2)	1	1(2)		6
Memory cycle time	120/160(200)	150/200	120/160(200)	ns	1,6
Subsystem controller	Yes	Yes	Yes		
Peripheral circuitry	None	None	None		
Parity	No	No	No		
Size (TRAM size)	2	2	2		
Length	3.66	3.66	3.66	inch	
Pitch between pins	3.30	3.30	3.30	inch	
Width	2.15	2.15	2.15	inch	
Component height above PCB	9.2	9.2	9.2	mm	2
Component height below PCB	3.7	3.7	3.7	mm	3
Weight	68	68	68	g	
Storage temperature	0-70	0-70	0-70	°C	
Operating temperature	0-50	0-50	0-50	°C	4
Power supply voltage (VCC)	4.75-5.25	4.75-5.25	4.75-5.25	Volt	
Power consumption	6	5	5	W	5

Table 21.5 IMS B404 specification

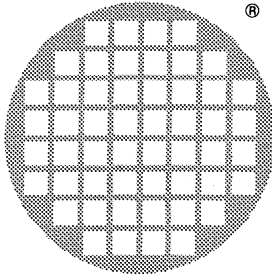
Notes:

- 1 The two figures quoted refer to (i) SRAM cycle time, and (ii) DRAM cycle time.
- 2 Since the IMS B404 makes use of 1 Mbit ZIP RAMs, this dimension is larger than is normally stated for TRAMs.
- 3 This dimension includes the thickness of the PCB.
- 4 The figure quoted refers to the ambient air temperature.
- 5 The power consumption is the worst case value obtained when a sample of IMS B404 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.
- 6 The B404-5 incorporates some wait-state logic that generates an extra memory cycle wait-state whenever the same bank of DRAM is accessed consecutively. The figures in parentheses refer to these instances.

21.1.9 Ordering Information

Description	Order Number
IMS B404 TRAM with IMS T800-25	IMS B404-5
IMS B404 TRAM with IMS T800-20	IMS B404-3
IMS B404 TRAM with IMS T425-25	IMS B404-8

Table 21.6 Ordering information



inmos[®]

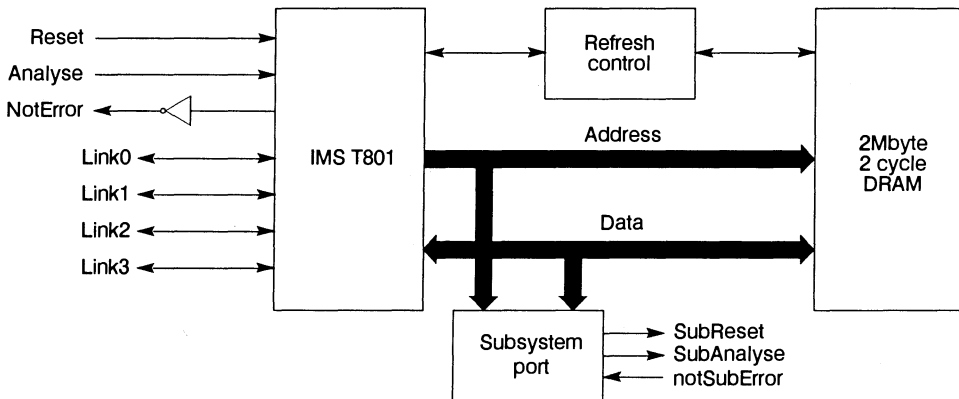
IMS B428

32-bit transputer

2 Mbytes

Size 2 TRAM

Engineering Data



FEATURES

- IMS T801 Transputer with non-multiplexed buses
- 2 Mbytes zero wait state DRAM at 25 MHz
- 2 cycle access (typical), 3 cycle access (worst case)
- DRAM refresh interleaved with processor memory accesses
- Size 2 TRAM
- Communicates via 4 INMOS serial links
- Package has only 16 active pins
- Sub-system port for resetting transputer networks
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The IMS B428 provides a new option to users wanting maximum performance from Transputer based systems. The IMS T801 Floating Point Transputer features separate address and data busses which allows memory cycles to be completed in two clock cycles. New generation non-multiplexed dynamic memories are used to take advantage of this high bandwidth (50 Mbytes/second) memory interface. This design avoids the memory bandwidth reduction normally associated with dynamic memory refresh by splitting the memory into two banks and performing a refresh operation on one bank while the processor accesses the other. The result is a compact unit with enough memory to run major applications significantly faster than designs using a multiplexed bus.

22.1 IMS B428 TRAM engineering data

22.1.1 Introduction

The IMS B428 is one of a range of INMOS TRAnsputer Modules (TRAMs) incorporating an IMS T801 transputer and 2Mbyte of dynamic RAM. In effect, these TRAMs are board level transputers with a simple, standardized interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with a minimum of design effort.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

22.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7
Subsystem Services			
SubSystemReset	out	Subsystem reset	1b
SubSystemAnalyse	out	Subsystem error analysis	1c
notSubSystemError	in	Subsystem error indicator	1a

Table 22.1 IMS B428 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in figure 22.3.

22.1.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached, but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. (The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

LinkSpeedA and LinkSpeedB (pins 6 and 7)

LinkSpeedA sets the speed of the transputer links 0–3. When this input is low, the links operate at 10 Mbits/s, and when high the links operate at 20 Mbits/s. **LinkSpeedB** is not connected.

Link signals

Whilst the links obey a protocol identical to that described in the *Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0–3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0–3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

22.1.4 Subsystem signals

The IMS B428 has a subsystem port in addition to the usual TRAM signals. This enables the TRAM to reset or analyse a subsystem of other TRAMs and/or motherboards. The polarity of these signals is the same as that of the **Reset**, **Analyse** and **notError** standard TRAM signals. Therefore, the IMS B428 subsystem can drive other TRAMs on the same motherboard with no intermediate logic. However, **SubSystemReset** and **SubSystemAnalyse** must go through inverting buffers if they are to drive a subsystem off the motherboard.

These subsystem signals are accessed by writing or reading to control registers in the transputer memory space. See section 22.1.5.

22.1.5 Memory configuration

The IMS B428 is able to access 2 Mbytes of memory. This is comprised of 4 Kbytes of internal transputer memory and 2Mbytes of DRAM. The memory is arranged as 2 banks, each of 1Mbyte. These banks are interleaved so that as successive words from memory are accessed, alternate banks are used.

The IMS B428 completes the majority of memory accesses in two cycles. The condition for wait-stating the processor is that a write or refresh has been performed on one of the two banks of memory and this cycle is then immediately followed by another access to that bank. If a delay of one or more clock cycles occurs after the write or refresh then the pending wait state is cancelled.

A consequence of this arrangement is that block move operations will vary in speed according to which banks the source and destination are in. If the source and destination are in the same bank then the block move will proceed with no wait states, otherwise 1 wait state will be added per transfer, giving 2.5 cycle operation.

Refresh to the DRAMs is performed in parallel with write cycles — while one bank is being written the other is refreshed if this is needed. A mechanism is provided to continue refresh during periods when the processor is not executing any write cycles to the memory.

Subsystem register locations

The subsystem register addresses start at hardware address #00000000 in all TRAMs that utilize a 32-bit processor, allowing software compatibility between TRAMs. These registers are located as shown in Table 22.2.

Register	Hardware byte address
SubSystemReset (write only)	#00000000
SubSystemAnalyse (write only)	#00000004
notSubSystemError (read only)	#00000000

Table 22.2 Subsystem address locations

Setting bit 0 in either the reset or the analyse registers asserts the corresponding signal. Similarly, clearing bit 0 deasserts the signal. When an error occurs in the subsystem, bit 0 of the error location becomes set.

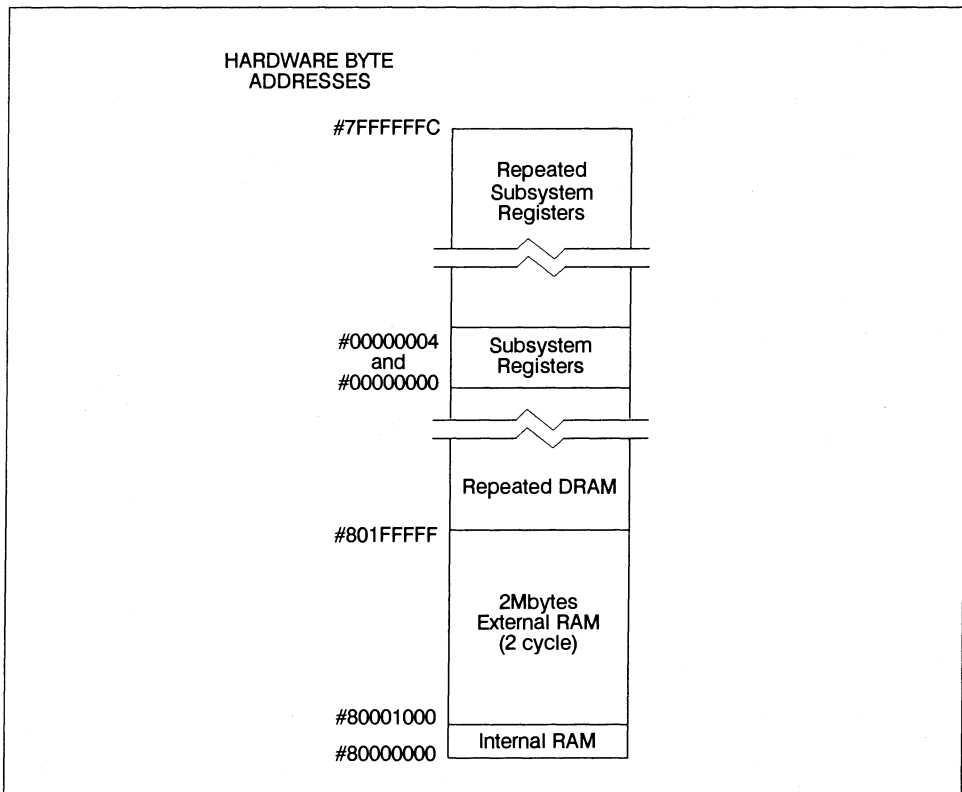


Figure 22.1 Memory map

22.1.6 Mechanical details

Figure 22.2 indicates the vertical dimensions of a single IMS B428 and figure 22.3 shows the outline drawing of the of the IMS B428.

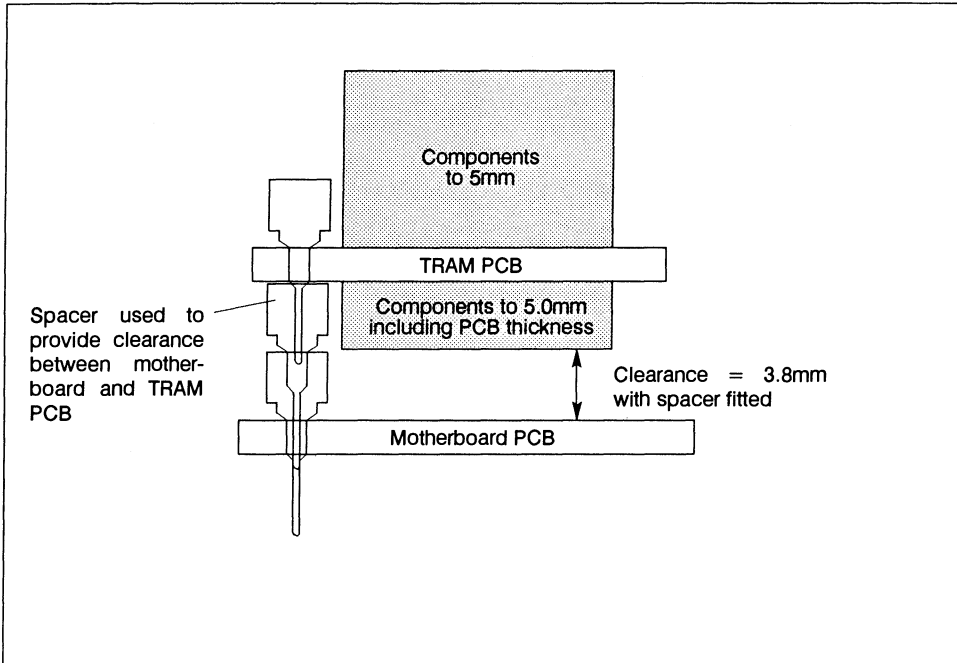


Figure 22.2 IMS B428 height specification

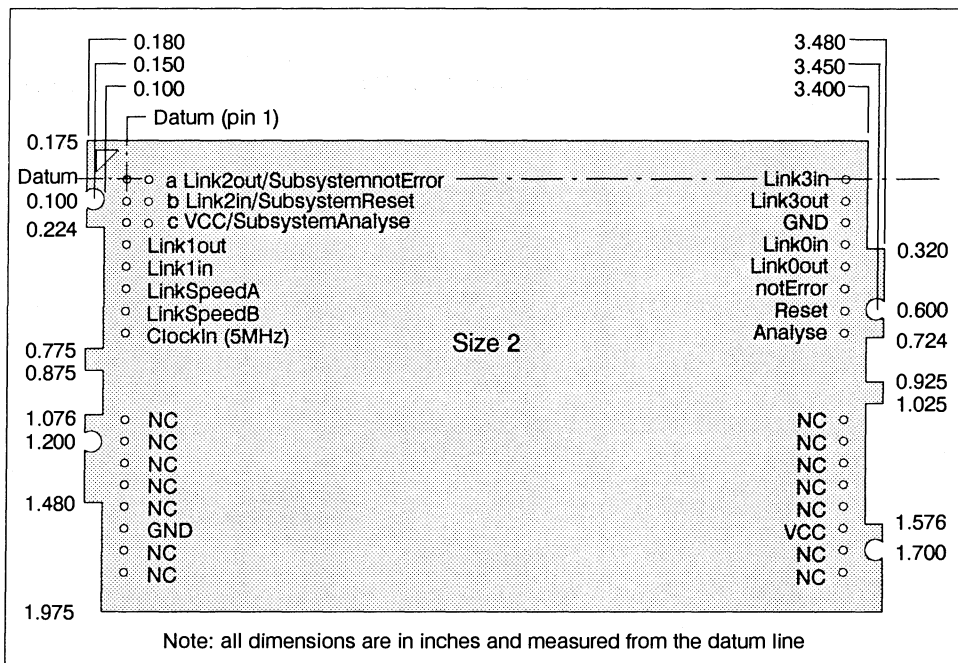


Figure 22.3 PCB profile drawing and pinout

22.1.7 Installation

Since the IMS B428 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B428 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. The height of components mounted on the underside of the IMS B428 dictate the use of these spacer strips to allow adequate clearance when the TRAM is mounted on a motherboard.

If the subsystem signals are required, plug a 3-way header strip into the solder-side sockets on the IMS B428. A 3-way spacer strip will also be needed as the TRAM will also be fitted with spacers.

Where the IMS B428 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B428 (see figure 22.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B428, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B428 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

22.1.8 Specification

TRAM feature		Unit	Notes
Transputer type	T801-25		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	None		
SRAM wait states	N/A		
Amount of DRAM	2	Mbyte	
DRAM wait states	0(1)		1
Memory cycle time	80	ns	
Subsystem controller	Yes		
Peripheral circuitry	None		
Memory parity	No		
TRAM size	2		
Length	3.66	inch	
Width	2.15	inch	
Pitch between pins	3.30	inch	
Component height above PCB	5.0	mm	
Component height below PCB	5.2	mm	2
Weight	70	g	3
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	
Power supply voltage (Vcc)	4.75-5.25	Volt	
Power consumption (Max)	10	W	
Power consumption (Typical)	6.5	W	

Table 22.3 IMS B428 specification

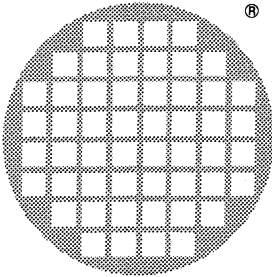
Notes

- 1 One wait state is inserted if a write to one of the two banks is immediately followed by another access to that same bank.
- 2 This dimension includes the thickness of the PCB.
- 3 Weight is approximate

22.2 Ordering Information

Description	Order Number
IMS B428 TRAM with IMS T801-25	IMS B428-12

Table 22.4 Ordering information



inmos[®]

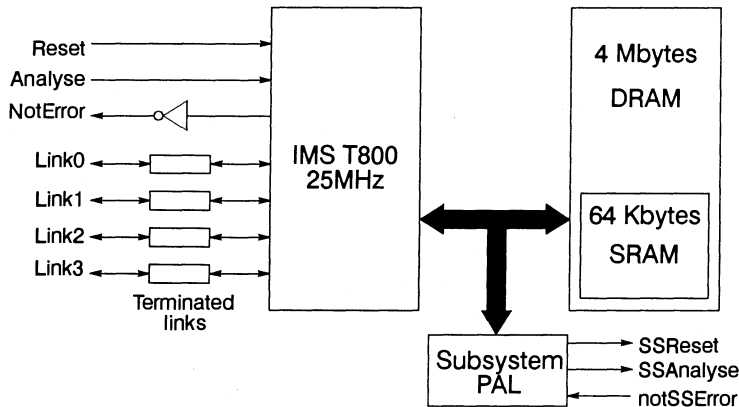
IMS B417 TRAM

32-bit transputer

4 Mbytes

Size 4

Engineering Data



FEATURES

- IMS T800 25 MHz Transputer
- 64 Kbytes of zero wait-state SRAM
- 4 Mbytes of single wait-state DRAM
- Subsystem controller circuitry
- Communicates via 4 INMOS serial links (Selectable between 10 or 20 Mbits/s)
- Package has only 16 active pins
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The IMS B417 uses the IMS T800 25MHz transputer. The 4 Mbytes of DRAM is sufficient to run the Ada compiler from Alsys. Also provided is 64 Kbytes of fast SRAM (3 cycle), so any technique which puts most frequently accessed memory locations near the bottom of memory will speed up the processing.

23.1 IMS B417 TRAM engineering data

23.1.1 Introduction

The IMS B417 is one of a range of INMOS TRANputer Modules (TRAMs) incorporating a IMS T800 transputer, 64 Kbytes of static RAM and 4 Mbytes of dynamic RAM. In effect, these TRAMs are board level transputers with a simple, standardized interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

23.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7
Subsystem Services			
SubSystemReset	out	Subsystem reset	1b
SubSystemAnalyse	out	Subsystem error analysis	1c
notSubSystemError	in	Subsystem error indicator	1a

Table 23.1 IMS B417 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Figure 23.3.

23.1.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached, but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. (The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

LinkSpeedA and LinkSpeedB (pins 6 and 7)

LinkSpeedA and **LinkSpeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low, the link(s) operate at 10 Mbits/s, and when high the link(s) operate at 20 Mbits/s.

Link signals

Whilst the links obey a protocol identical to that described in *The Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

23.1.4 Subsystem signals

The IMS B417 has a subsystem port in addition to the usual TRAM signals. This enables the TRAM to reset or analyse a subsystem of other TRAMs and/or motherboards. The polarity of these signals is the same as that of the **Reset**, **Analyse** and **notError** standard TRAM signals. Therefore, the IMS B417 subsystem can drive other TRAMs on the same motherboard with no intermediate logic. However, **SubSystemReset** and **SubSystemAnalyse** must go through inverting buffers if they are to drive a subsystem off the motherboard.

These subsystem signals are accessed by writing or reading to control registers in the transputer memory space.

23.1.5 Memory configuration

The IMS B417 is able to access 4 Mbytes of memory. This is comprised of 4 Kbytes of internal transputer memory, 60 Kbytes of external SRAM and 4032 Kbytes of external DRAM. There are, in fact, 64 Kbytes of SRAM components and 4 Mbytes of DRAM components on the board, but the address spaces of each type of memory are superimposed. Therefore, the total memory available is limited to 4 Mbytes.

Location of external memory

Tables 23.2 and 23.3 show the start addresses of the different types of external memory on the IMS B417 (the '#' sign indicates a hexadecimal number). The internal RAM on the IMS T800 occupies the first 4 Kbytes of address space.

	Hardware byte address
From:	#80001000
To:	#8000FFFF

Table 23.2 Location of external SRAM on the IMS B417

	Hardware byte address
From:	#80010000
To:	#803FFFFF

Table 23.3 Location of external DRAM on the IMS B417

Since the internal memory on the IMS T800 is 1 cycle, the external SRAM is 3 cycle and the DRAM is 4 cycle, a memory speed hierarchy is established. This architecture allows programmers to structure their code for optimum performance.

Subsystem register locations

The subsystem register addresses start at hardware address #00000000 in all TRAMs that utilize a 32-bit processor, allowing software compatibility between TRAMs. These registers are located as shown in Table 23.4.

Register	Hardware byte address
SubSystemReset (write only)	#00000000
SubSystemAnalyse (write only)	#00000004
notSubSystemError (read only)	#00000000

Table 23.4 Subsystem address locations

Setting bit 0 in either the reset or the analyse registers asserts the corresponding signal. Similarly, clearing bit 0 deasserts the signal. When an error occurs in the subsystem, bit 0 of the error location becomes set.

Byte locations #00000008 and #0000000C are unused. The subsystem registers are repeated at every sixteenth byte location in the positive address space. See Figure 23.1.

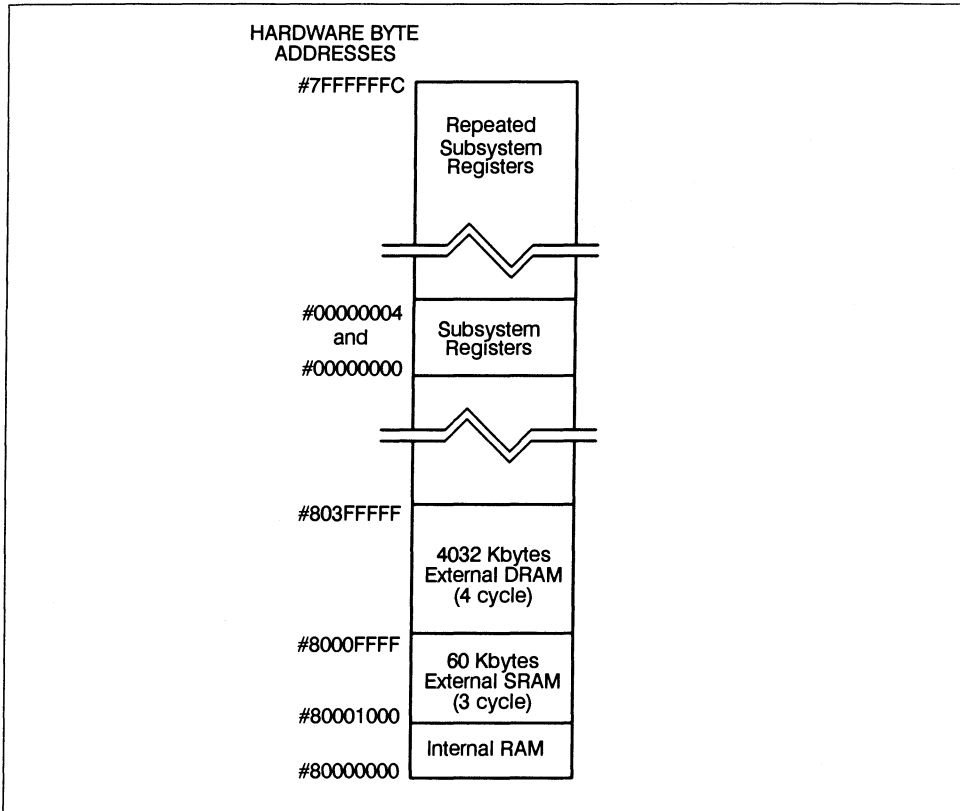


Figure 23.1 Memory map

23.1.6 Mechanical details

Figure 23.2 indicates the vertical dimensions of a single IMS B417 and Figure 23.3 shows the outline drawing of the IMS B417.

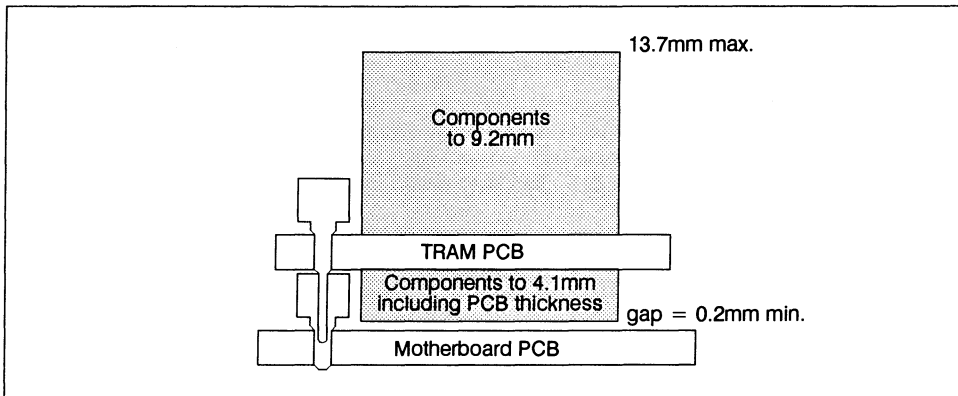


Figure 23.2 IMS B417 height specification

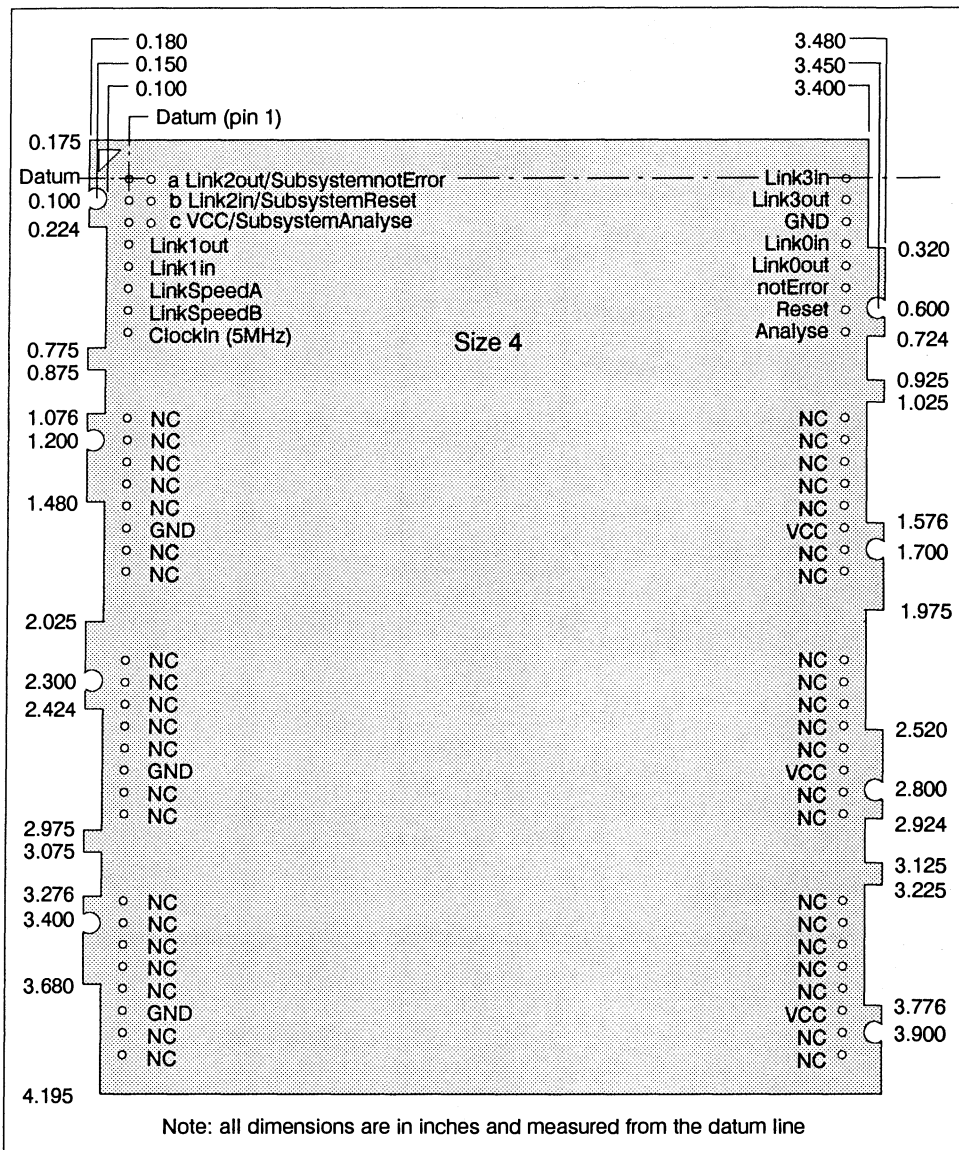


Figure 23.3 IMS B417 outline drawing

23.1.7 Installation

Since the IMS B417 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B417 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

If the subsystem signals are required, plug a 3-way header strip into the solder-side sockets (aside pins 1-3) on the IMS B417.

Plug the IMS B417 into the motherboard. Where the IMS B417 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B417 (see Figure 23.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B417, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B417 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

23.1.8 Specification

TRAM feature	IMS B417-5	Unit	Notes
Transputer type	IMS T800-25		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	64	Kbyte	
SRAM "wait states"	0		
SRAM cycle time	120	ns	
Amount of DRAM	4	Mbyte	
DRAM "wait states"	1		
DRAM cycle time	160	ns	
Subsystem controller	Yes		
Peripheral circuitry	None		
Parity	No		
Size (TRAM size)	4		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	4.35	inch	
Component height above PCB	9.2	mm	
Component height below PCB	3.7	mm	1
Weight	110	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	6	W	3

Table 23.5 IMS B417 specification

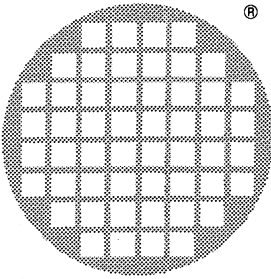
Notes:

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B417 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

23.1.9 Ordering Information

Description	Order Number
IMS B417 TRAM with IMS T800-25	IMS B417-5

Table 23.6 Ordering information

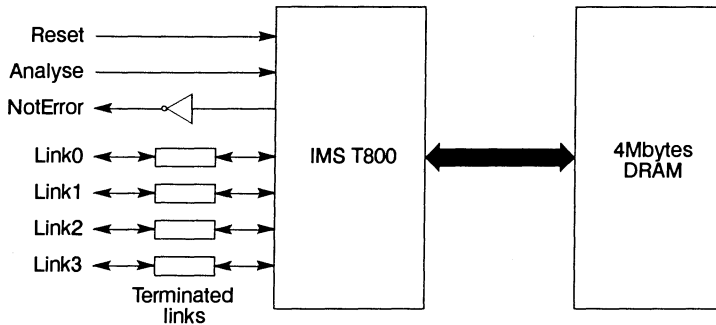


inmos®

IMS B426

32-bit transputer
4 Mbytes
Size 1 TRAM

Engineering Data



FEATURES

- IMS T800 Transputer
- 4 Mbyte of one wait-state DRAM (160 ns memory cycle time)
- Size 1 TRAM
- Communicates via 4 INMOS serial links
- Package has only 16 active pins
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The IMS B426 TRAM is the ideal module for applications where space is at a premium. With 4 Mbyte of DRAM on a size 1 TRAM the following configurations are possible:

- 8 transputers and 32 Mbytes of memory can be installed in a single 6U VMEbus slot (using the IMS B014 motherboard)
- 4 transputers and 16 Mbytes of memory (using an IMS B017 PS/2 motherboard).

* For IBM PC read: original PC, XT, AT, PS/2 Model 30 and most clones.

24.1 Description

The IMS B426 is an INMOS TRAnsputer Module (TRAM) incorporating an IMS T800 transputer and 4 Mbytes of dynamic RAM.

TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort. TRAMs may be plugged into motherboards, which provide the necessary electrical signals, mechanical support and usually, an interface to a host machine. A variety of motherboards are now available from INMOS and from third-party vendors for most of the common computing platforms.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

24.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7

Table 24.1 IMS B426 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in figure 24.3.

24.3 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

24.3.1 notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. (The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

24.3.2 LinkSpeedA and LinkSpeedB (pins 6 and 7)

LinkSpeedA and **LinkSpeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low the link(s) operate at 10 Mbits/s and when high the link(s) operate at 20 Mbits/s.

24.3.3 Link signals

Whilst the links obey a protocol identical to that described in the *Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

24.4 Memory configuration

The internal RAM of the IMS T800 occupies the first 4 Kbytes of address space. The next 4 Mbytes is occupied by the external dynamic RAM present on the TRAM. The external RAM is repeated in 4 Mbyte blocks throughout the higher address space.

Table 24.2 details the start and end addresses of the external memory and figure 24.1 shows a graphical representation of the memory map (the “#” sign indicating a hexadecimal number).

Hardware byte address	
From:	#80001000
To:	#80400FFF

Table 24.2 Location of external memory on the IMS B426

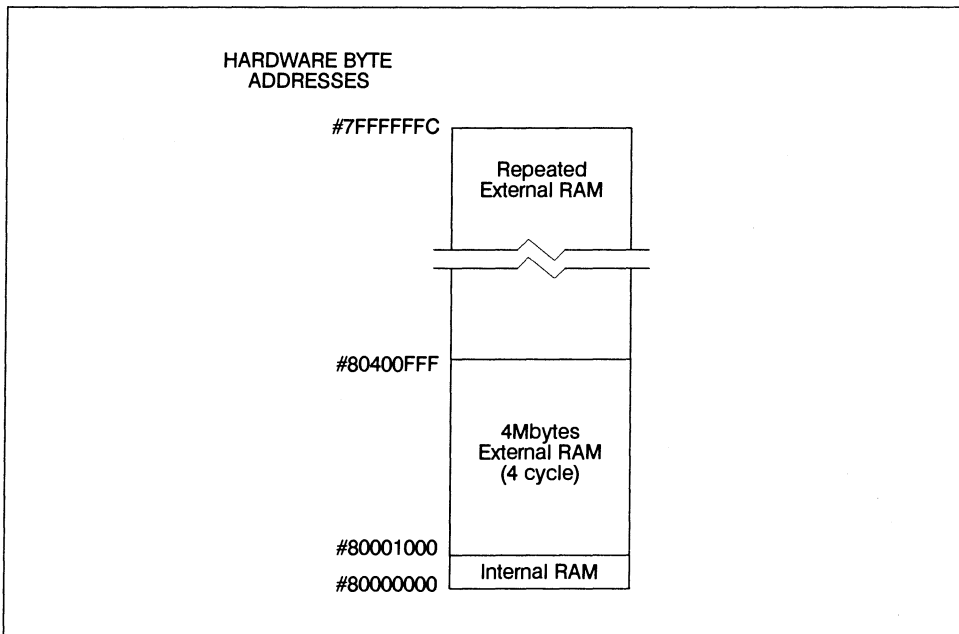


Figure 24.1 Memory map

24.5 Mechanical details

Figure 24.2 indicates the vertical dimensions of a single IMS B426 TRAM and Figure 24.3 shows the outline drawing of the IMS B426.

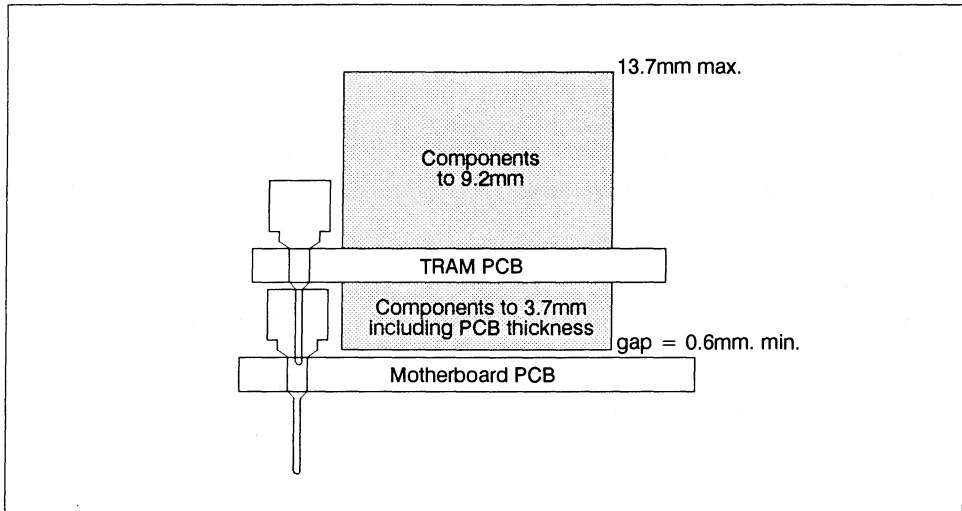


Figure 24.2 IMS B426 height specification

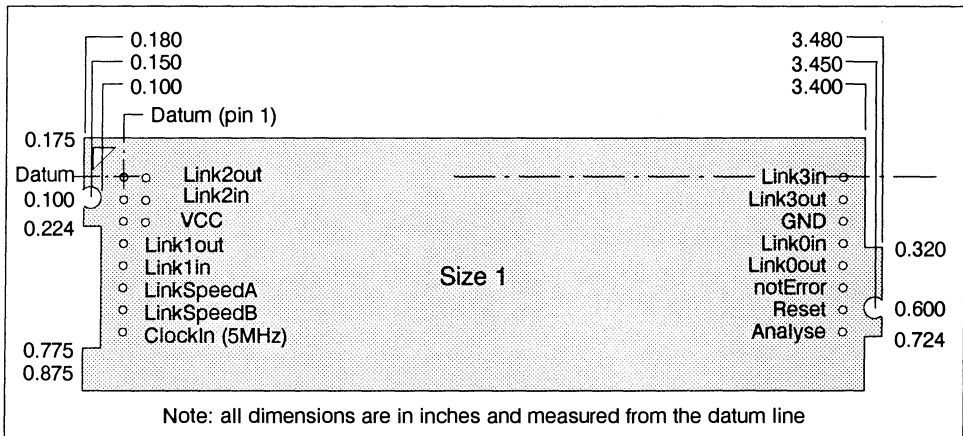


Figure 24.3 PCB profile drawing and pinout

24.6 Installation

Since the IMS B426 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B426 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B426 carefully into the motherboard. Where the IMS B426 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B426 (see figure 24.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot. If it is envisaged that the assembly is likely to be subjected to any vibrations, it is recommended that the TRAM is secured to the motherboard using nylon M3 nuts and bolts. The bolts should be inserted through the fixing holes on the motherboard, and through the castillations on two edges of the TRAM. A number of these nuts and bolts are supplied with each of the INMOS motherboards.

Should it be necessary to unplug the IMS B426, it is advised that, having removed any retaining nuts and bolts, it is gently levered out while keeping it as flat as possible. As soon as the IMS B426 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

24.7 Specification

TRAM feature	IMS B426-5	Unit	Notes
Transputer type	T800-25		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	None		
SRAM 'wait states'	N/A		
Amount of DRAM	4	Mbyte	
DRAM 'wait states'	1		
Memory cycle time	160	ns	
Subsystem controller	No		
Peripheral circuitry	None		
Parity	No		
Size (TRAM size)	1		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	1.05	inch	
Component height above PCB	9.2	mm	
Component height below PCB	3.7	mm	1
Weight	50	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	4	W	3

Table 24.3 IMS B426 specification

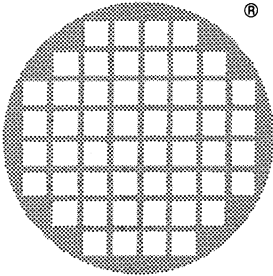
Notes

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B426 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

24.8 Ordering Information

Description	Order Number
IMS B426 TRAM with IMS T800-25	IMS B426-5

Table 24.4 Ordering information

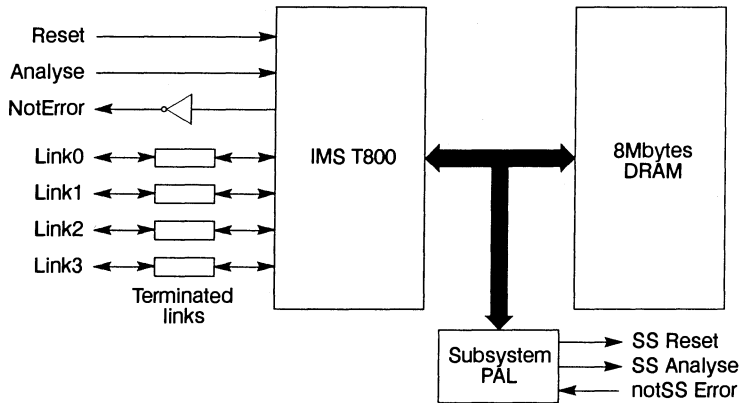


inmos[®]

IMS B427

32-bit transputer
8 Mbytes
Size 2 TRAM

Engineering Data



FEATURES

- 25MHz IMS T800 Transputer
- 8 Mbyte of one wait-state DRAM (160 ns memory cycle time)
- Size 2 TRAM
- Communicates via 4 INMOS serial links
- Package has only 16 active pins
- Subsystem control circuitry
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The IMS B427 is a compact size 2 TRAM offering 8Mbytes of 4-cycle DRAM and subsystem controller circuitry.

With a large amount of external memory, the B427 is able to run all of the INMOS development tools including the ADA compiler from Alsys. It is ideally suited for applications using large amounts of memory, allowing programs such as simulation and AI evaluation to run quickly and efficiently.

25.1 Description

The IMS B427 is an INMOS TRANputer Module (TRAM) incorporating an IMS T800 transputer, 8 Mbytes of dynamic RAM and subsystem control circuitry.

TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort. TRAMs may be plugged into motherboards, which provide the necessary electrical signals, mechanical support and usually, an interface to a host machine. A variety of motherboards are now available from INMOS and from third-party vendors for most of the common computing platforms.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

25.1.1 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkspeedA,B	in	Transputer link speed selection	6,7
Subsystem services			
Subsystem reset	out	Subsystem reset	1b
Subsystem analyse	out	Subsystem error analysis	1c
Subsystem error	in	Subsystem error indicator	1a

Table 25.1 IMS B427 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in figure 25.3.

25.1.2 Standard TRAM signals

A TRAM can be regarded as a transputer with extra RAM attached but with only 16 signals brought out to the TRAM pins. The majority of the TRAM pins function in exactly the same way as the corresponding transputer signals, which are detailed in *The Transputer Databook*. However, a few of these signals are slightly different from the transputer specification as follows:

25.2 notError (pin 11)

This is an open collector signal. It is driven low when there is an error; otherwise it is pulled high by a resistor on the motherboard. This enables the **notError** outputs on several TRAMs to be wire-ORed together. (The TRAM specification recommends that no more than 10 **notError** outputs are connected together).

25.3 LinkSpeedA and LinkSpeedB (pins 6 and 7)

LinkSpeedA and **LinkSpeedB** set the speed of transputer link 0 and links 1-3 respectively. When the appropriate input is low the link(s) operate at 10 Mbits/s and when high the link(s) operate at 20 Mbits/s.

25.4 Link signals

Whilst the links obey a protocol identical to that described in the *Transputer Databook*, there are some differences in the electrical characteristics.

LinkIn0-3 The link inputs have pull-down resistors to ensure that they are disabled when they are not connected. Diodes are also included for protection against electrostatic discharge.

LinkOut0-3 The link outputs have resistors connected in series for matching to a 100 ohm transmission line.

25.5 Subsystem signals

The IMS B427 has a subsystem port in addition to the usual TRAM signals. This enables the TRAM to reset or analyse a subsystem of other TRAMs and/or motherboards. The polarity of these signals is the same as that of the **Reset**, **Analyse** and **notError** standard TRAM signals. Therefore, the IMS B427 subsystem can drive other TRAMs on the same motherboard with no intermediate logic. However, **SubSystemReset** and **SubSystemAnalyse** must go through inverting buffers if they are to drive a subsystem off the motherboard.

These subsystem signals are accessed by writing or reading to control registers in the transputer memory space.

25.5.1 Memory configuration

The internal RAM of the IMS T800 occupies the first 4 Kbytes of address space. The next 8 Mbytes is occupied by the external dynamic RAM present on the TRAM.

Table 25.2 details the start and end addresses of the external memory and figure 25.1 shows a graphical representation of the memory map (the “#” sign indicating a hexadecimal number).

	Hardware byte address
From:	#80001000
To:	#807FFFFF

Table 25.2 Location of external memory on the IMS B427

Subsystem register locations

The subsystem register addresses start at hardware address #00000000 in all TRAMs that utilize a 32-bit processor, allowing software compatibility between TRAMs. These registers are located as shown in Table 25.3.

Register	Hardware byte address
SubSystemReset (write only)	#00000000
SubSystemAnalyse (write only)	#00000004
notSubSystemError (read only)	#00000000

Table 25.3 Subsystem address locations

Setting bit 0 in either the reset or the analyse registers asserts the corresponding signal. Similarly, clearing bit 0 deasserts the signal. When an error occurs in the subsystem, bit 0 of the error location becomes set.

Byte locations #00000008 and #0000000C are unused. The subsystem registers are repeated at every sixteenth byte location in the positive address space. See Figure 25.1.

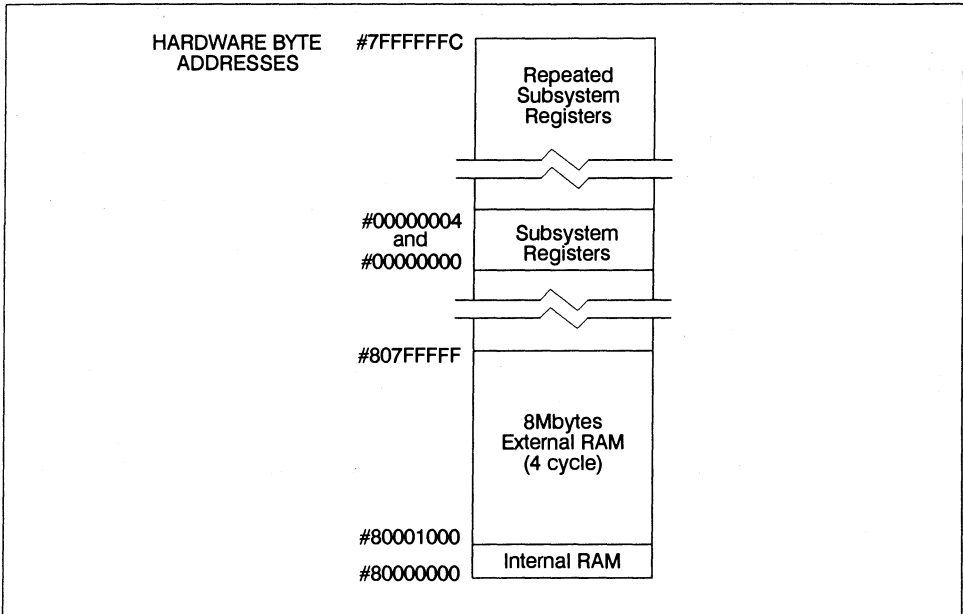


Figure 25.1 Memory map

25.5.2 Mechanical details

Figure 25.2 indicates the vertical dimensions of a single IMS B427 TRAM and Figure 25.3 shows the outline drawing of the IMS B427.

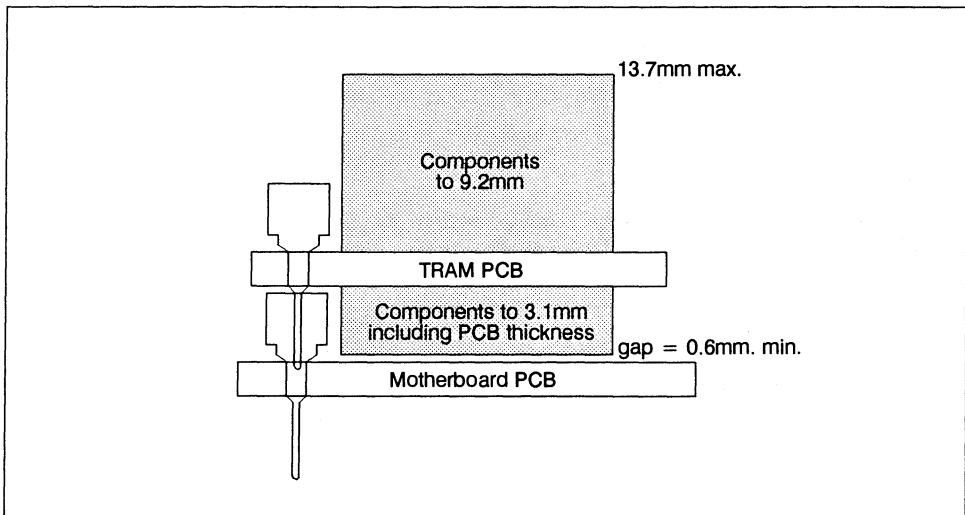


Figure 25.2 IMS B427 height specification

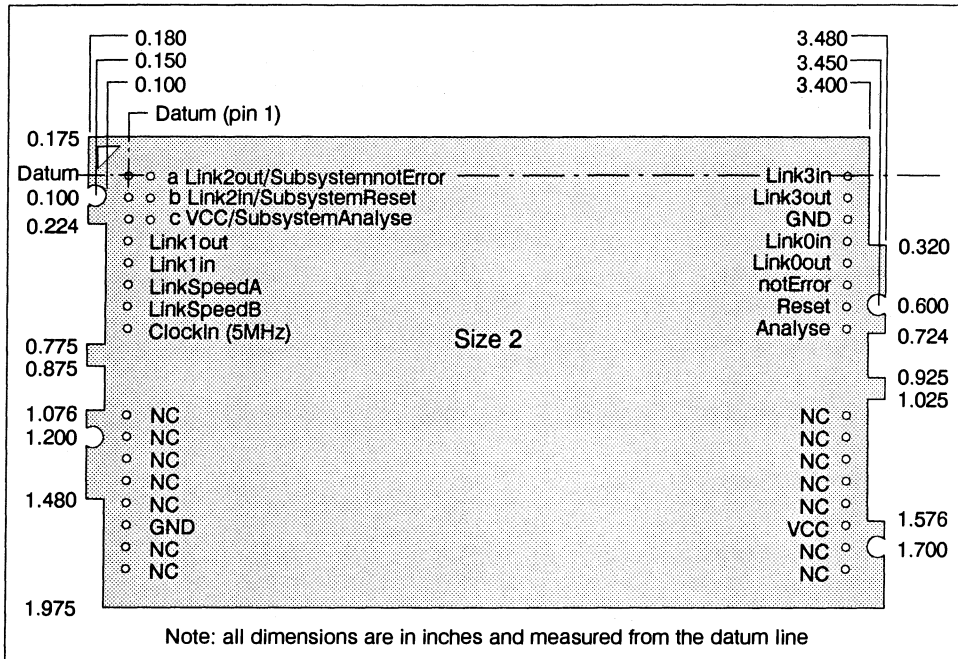


Figure 25.3 PCB profile drawing and pinout

25.5.3 Installation

Since the IMS B427 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B427 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B427 carefully into the motherboard. Where the IMS B427 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B427 (see figure 25.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot. If it is envisaged that the assembly is likely to be subjected to any vibrations, it is recommended that the TRAM is secured to the motherboard using nylon M3 nuts and bolts. The bolts should be inserted through the fixing holes on the motherboard, and through the castlans on two edges of the TRAM. A number of these nuts and bolts are supplied with each of the INMOS motherboards.

Should it be necessary to unplug the IMS B427, it is advised that, having removed any retaining nuts and bolts, it is gently levered out while keeping it as flat as possible. As soon as the IMS B427 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

25.5.4 Specification

TRAM feature	IMS B427-5	Unit	Notes
Transputer type	T800-25		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	None		
SRAM 'wait states'	N/A		
Amount of DRAM	8	Mbyte	
DRAM 'wait states'	1		
Memory cycle time	160	ns	
Subsystem controller	Yes		
Peripheral circuitry	None		
Parity	No		
Size (TRAM size)	2		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	2.15	inch	
Component height above PCB	9.2	mm	
Component height below PCB	3.1	mm	1
Weight	63	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	4.6	W	3

Table 25.4 IMS B427 specification

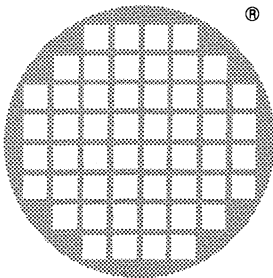
Notes

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B427 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

25.5.5 Ordering Information

Description	Order Number
IMS B427 TRAM with IMS T800-25	IMS B427-5

Table 25.5 Ordering information



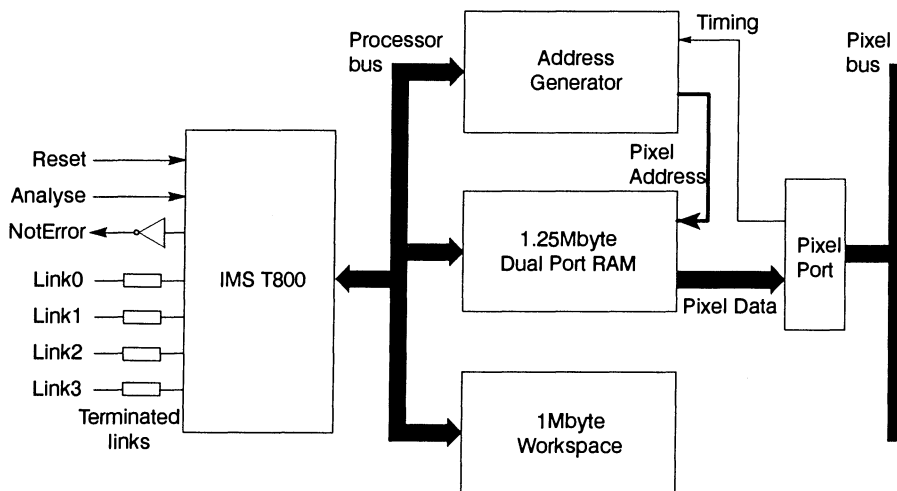
inmos[®]

IMS B408

Frame store TRAM

Size 8

Engineering Data



FEATURES

- IMS T800 Transputer
- 1 Mbytes single wait-state work space DRAM
- 1.25 Mbytes single wait-state dual port DRAM
- Dual Port supports continuous data rates up to 100 Mbytes/s
- Communicates via 4 INMOS serial links (Selectable between 10 or 20 Mbits/s)
- Designed to a published specification (*INMOS Technical Note 29*)

GENERAL DESCRIPTION

The IMS B408 implements the drawing and image storage parts of a medium to high performance graphics system. It incorporates a powerful 32-bit microprocessor with on-chip FPU, 1 Mbyte of workspace RAM and 1.25 Mbyte of display RAM accessible to the processor and dual ported to the Pixel Port. The pixel port is capable of sustaining continuous data transmission at up to 100 Mbytes/sec, independently of the processor, and under control of an autonomous address generator. The IMS B408 supports both interlaced and non-interlaced displays of arbitrary resolution up to 1024 × 768 pixels. At lower resolutions multiple frame buffers are supported; e.g. 4 frame buffers of 640 × 480 pixels.

26.1 IMS B408 TRAM engineering data

26.1.1 Introduction

The IMS B408 is one of a range of INMOS TRANsputer Modules (TRAMs). In effect, TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort¹.

The IMS B408 is designed to be used with the IMS B409 display driver TRAM. When connected to an IMS B409 via the INMOS Pixel Bus (and a suitable video monitor) a complete drawing and display system is formed. System performance is increased simply by adding more IMS B408s.

The IMS B408 performs the drawing function in such a system. The graphics processor is an IMS T800; a fast 32 bit processor with on-chip FPU. Image data is drawn into 1.25 Mbyte of dual port RAM; a further 1 Mbyte of RAM is provided for program/data storage. Image data is output through the pixel bus pixel port under the control of the dual port address generator. The address generator is programmable and responds to system timing signals from the pixel bus.

26.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkSpeedA,B	in	Transputer link speed selection	6,7

Table 26.1 IMS B408 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Fig.26.4.

LinkOut0-3 Transputer link output signals. These outputs are intended to drive into transmission lines with a characteristic impedance of 100Ω. They can be connected directly to the **LinkIn** pins of other transputers or TRAMs.

LinkIn0-3 Transputer link input signals. These are the link inputs of the transputer. Each input has a 10KΩ resistor to **GND** to establish the idle state, and a diode to **VCC** as protection against ESD. They can be connected directly to the **LinkOut** pins of other transputers or TRAMs.

LinkSpeedA, LinkSpeedB These select the speeds of **Link0** and **Link1,2,3** respectively. Table 26.2 shows the possible combinations.

¹ Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Dual-In-Line Transputer Modules (TRAMs)* and *Module Motherboard Architecture* which are included later in this databook. The *Transputer Databook* may also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

LinkSpeedA	LinkSpeedB	Link0	Link1,2,3
0	0	10 Mbits/s	10 Mbits/s
0	1	10 Mbits/s	20 Mbits/s
1	0	20 Mbits/s	10 Mbits/s
1	1	20 Mbits/s	20 Mbits/s

Table 26.2 Link speed selection

ClockIn A 5MHz input clock for the transputer. The transputer synthesises its own high frequency clocks. **ClockIn** should have a stability over time and temperature of 200ppm. **ClockIn** edges should be monotonic within the range 0.8V to 2.0V with a rise/fall time of less than 8ns.

Reset Resets the transputer, and other circuitry. **Reset** should be asserted for a minimum of 100ms. After **Reset** is deasserted a further 100ms should elapse before communication is attempted on any link. After this time, the transputer on this TRAM is ready to accept a boot packet on any of its links.

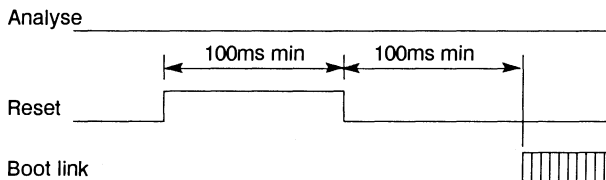


Figure 26.1 Reset timing

Analyse is used, in conjunction with **Reset**, to stop the transputer. It allows internal state to be examined so that the cause of an error may be determined. **Reset** and **Analyse** are used as shown in figure26.1. A processor in analyse mode can be interrogated on any of its links.

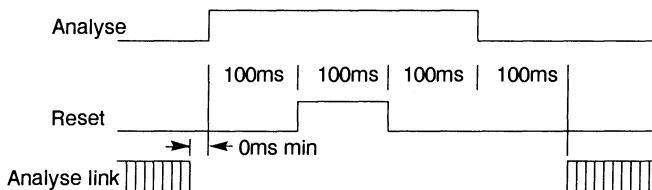


Figure 26.2 Analyse timing

notError An open collector output which is pulled low when the transputer asserts its Error pin. **notError** should be pulled high by a 10KΩ resistor to **VCC**. Up to 10 **notError** signals can be wired together. The combined error signal will be low when any of the contributing signals is low.

26.1.3 Pixel Port signals

The IMS B408 has a pixel data port in addition to the usual TRAM signals. This enables the TRAM to connect via the INMOS pixel bus to an IMS B409 display TRAM; and possibly several other IMS B408s. The pinout is defined in Table 26.3. The pixel bus uses 60-way IDC connectors and flat ribbon cable.

Pin No.	In/Out	Pin	Pin	In/Out	Pin No.
1		GND	D0	out	2
3	out	D1	GND		4
5	out	D2	D3	out	6
7		GND	D4	out	8
9	out	D5	GND		10
11	out	D6	D7	out	12
13		GND	D8	out	14
15	out	D9	GND		16
17	out	D10	D11	out	18
19		GND	D12	out	20
21	out	D13	GND		22
23	out	D14	D15	out	24
25		GND	D16	out	26
27	out	D17	GND		28
29	out	D18	D19	out	30
31		GND	D20	out	32
33	out	D21	GND		34
35	out	D22	D23	out	36
37		GND	D24	out	38
39	out	D25	GND		40
41	out	D26	D27	out	42
43		GND	D28	out	44
45	out	D29	GND		46
47	out	D30	D31	out	48
49		GND	notSEQclk	in	52
51		GND	notRAMclk	in	50
53		GND	notFieldSync	in	54
55		GND	notEarlyBlank	in	56
57		GND	notEvenField	in	58
59		GND	SysReady	in/out	60

Table 26.3 Pixel Bus pin designations

D0 – 31 Pixel data is output on a 32 bit bus. Transitions occur on the falling edge of **notSEQclk**. The data bus is open collector and carries inverted data. This allows data from different serial port modules to be ORed on the Pixel Bus.

notSEQclk A continuous input clock used as the timing reference by the dual port address generator. It has a maximum frequency of 25MHz. Data and control strobes transitions are synchronised to the falling edge of **notSEQclk**.

notRAMclk Used to clock pixel data from the dual port RAM onto the pixel bus. It is the same frequency and phase as **notSEQclk** but is gated so that it does not run during blanking. Thus, no data is lost during blanking. In the off state it is high.

notEarlyBlank A time-advanced version of the display blanking signal. Used by the dual port address generator as an early warning of when pixel data will be required and of when it should be turned off.

notFieldSync Resets the dual port address generator at the start of each field. Low during field fly-back (vertical blanking).

notEvenField Used by the dual port address generator to ensure that the pixel data for the correct display field is output when generating an interlaced display. A low on this signal indicates the even field of the odd/even field pair making up an interlaced frame.

SysReady This acts as a synchronisation mechanism for multiple modules. The IMS B408 has a writeable READY bit which drives an open collector output onto this wire; it also monitors its state. Only when all IMS B408s in a system have written 1 (ready) to their READY bits will SYSREADY be 1. This can be used to EVENT (interrupt) the IMS T800.

Electrical Specification

The open collector drivers used for the data bus are capable of sinking 64mA and must be pulled up by an external resistor network. This network is part of the pixel data input structure on the IMS B409.

The clock and control inputs have 4K7 Ω pull up resistors to establish an idle condition on each input when the bus is disconnected.

26.1.4 Memory Map

There are 2304 Kbytes of memory. This is comprised of 4 Kbytes of internal transputer memory and 2300 Kbytes of external DRAM. The upper 1280 Kbytes is dual ported to the pixel port. The lower 1024 Kbytes would normally be used for program storage and the dual ported area as a drawing area (frame buffer). Table 26.4 shows how the memory is mapped into the address space of the IMS T800 (the “#” sign indicates a hexadecimal number).

	Byte address	Cycle Time
IMS T800 on chip RAM	#80000000 - #80000FFF	50ns
External Workspace RAM	#80001000 - #800FFFFF	200ns
Dual port RAM	#80100000 - #8023FFFF	200ns

Table 26.4 Memory map of the IMS B408

26.1.5 Pixel Port control registers

There are a small number of control registers associated with the pixel port and its address generator. These registers are located as shown in Table 26.5.

Register	Byte address
Display Start (write only)	#00000000
Ready (read,write)	#00040000
SysReady (read only)	#00080000
Interlace Enable (read,write)	#000C0000
Event Mode (read,write)	#00100000
Output Enable (read,write)	#00140000

Table 26.5 Control register locations

Display Start This registers holds the address of the pixel at the top left hand corner of the displayed image. It can be used to implement flipping between multiple drawing buffers. Buffers must start on 64 kbyte boundaries.

Interlace Enable Selects an interlaced or non-interlaced display. Writing 1 causes the address generator to produce interlace addressing; writing 0 causes it to produce non-interlaced addressing.

Event Mode Selects the EVENT (interrupt) source to be either FieldSync or SysReady.

Output Enable Enables and disables the pixel port output buffers. Writing 1 enables the data output buffers; writing 0 disables them.

Ready Writing 0 drives SysReady low; writing 1 allows it to be pulled high.

SysReady is a read only location which reflects the condition of the SysReady wire. Bit 0 is read as 0 if SysReady is low; 1 if SysReady is high.

26.1.6 Mechanical details

Figure 26.3 indicates the vertical dimensions of a single IMS B408 and Figure 26.4 shows the outline drawing of the IMS B408. Note that the component height includes the height taken up by a cable plugged into the pixel port connector. This means that the IMS B408 on a motherboard occupies more than one card slot in a 0.8in pitch card cage.

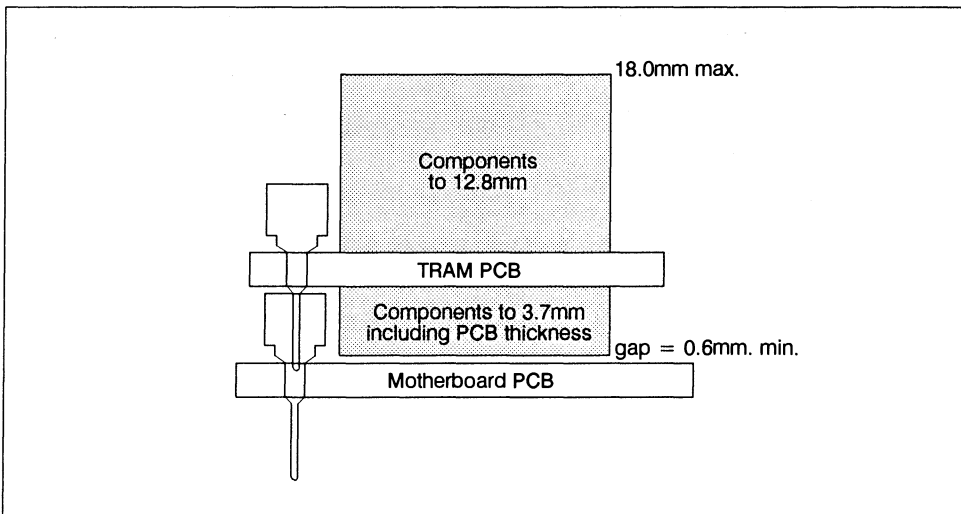


Figure 26.3 IMS B408 height specification

26.1.7 Installation

Since the IMS B408 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B408 may be supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B408 into the motherboard. Where the IMS B408 is being used with an INMOS motherboard, the copper triangle marking pin 1 on the IMS B408 (see Figure 26.4) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B408, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B408 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

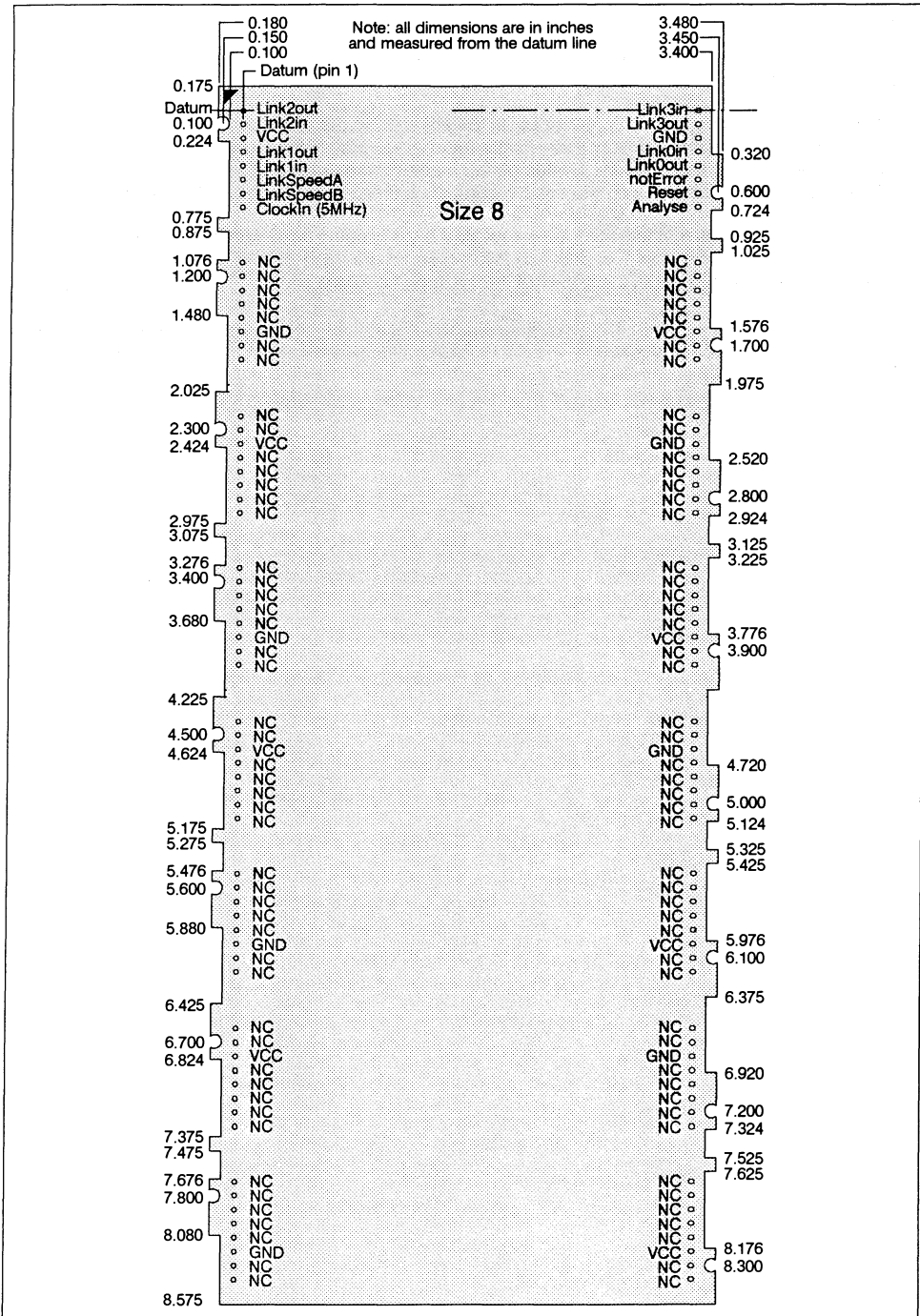


Figure 26.4 IMS B408 outline drawing (All dimensions in inches)

26.1.8 Specification

TRAM feature		Unit	Notes
Transputer type	IMS T800-20		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of DRAM	2.25	Mbyte	
DRAM "wait states"	1		
Memory cycle time	200	ns	
Subsystem controller	No		
Peripheral circuitry	Pixel Port		
Parity	No		
Size (TRAM size)	8		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	8.75	inch	
Component height above PCB	12.8	mm	1
Component height below PCB	3.0	mm	2
Weight	215	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	3
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	18	W	4

Table 26.6 IMS B408 specification

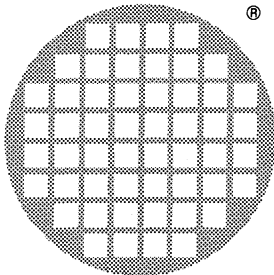
Notes:

- 1 This dimension is larger than is normally stated for TRAMs because of the requirement to connect to the pixel bus.
- 2 This dimension includes the thickness of the PCB.
- 3 The figure quoted refers to the ambient air temperature.
- 4 The power consumption is the worst case value obtained when a sample of IMS B408 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

26.1.9 Ordering Information

Description	Order Number
IMS B408 TRAM with IMS T800-20	IMS B408-3

Table 26.7 Ordering Information



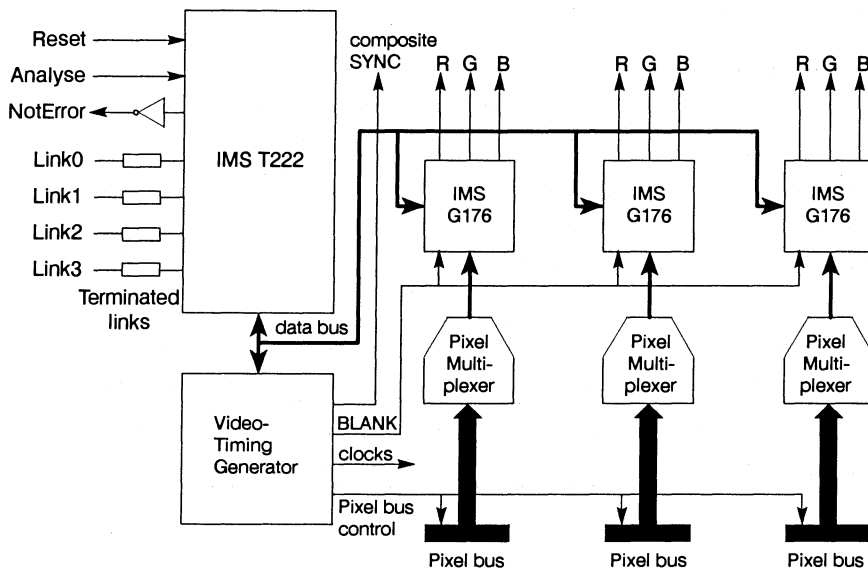
inmos[®]

IMS B409

Display TRAM

Size 8

Engineering Data



FEATURES

- IMS T222, 16-bit Transputer
- Video timing generator
- Pixel rates up to 64 MHz
- 8 or 18 bit pixels
- 3 IMS G176 colour look-up tables
- Designed to a published specification (*INMOS Technical Note29*)

GENERAL DESCRIPTION

The IMS B409 implements the timing generation and display driver parts of a medium to high performance graphics system. It consists of three pixel channels and a programmable video timing generator (VTG), controlled by an IMS T222. Each pixel channel consists of a 4-1 byte multiplexer and an IMS G176 colour look-up table (CLUT). Input to each pixel channel is by a separate pixel bus input and each channel generates a set of RGB outputs. The IMS B409 supports both interlaced and non-interlaced displays of arbitrary resolution up to a maximum dot rate of 64MHz.

27.1 IMS B409 TRAM engineering data

27.1.1 Introduction

The IMS B409 is one of a range of INMOS TRAnsputer Modules (TRAMs). In effect TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort¹.

The IMS B409 is designed to be used in conjunction with one or more IMS B408 frame store TRAMs. When connected to an IMS B408 via the INMOS Pixel Bus (and a suitable video monitor) a complete drawing and display system is formed. System performance is increased simply by adding more IMS B408s.

The IMS B409 has three pixel channels. Each channel inputs a 32 bit wide pixel stream from an INMOS Pixel Bus and processes it into a form suitable for display by a colour monitor. The IMS B409 also generates system timing and control signals and outputs them on each pixel bus.

27.1.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5 MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkSpeedA,B	in	Transputer link speed selection	6,7

Table 27.1 IMS B409 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in Fig 27.4.

LinkOut0-3 Transputer link output signals. These outputs are intended to drive into transmission lines with a characteristic impedance of 100Ω. They can be connected directly to the **LinkIn** pins of other transputers or TRAMs.

LinkIn0-3 Transputer link input signals. These are the link inputs to the transputer. Each input has a 10kΩ resistor to **GND** to establish the idle state, and a diode to **VCC** as protection against ESD. They can be connected directly to the **LinkOut** pins of other transputers or TRAMs.

LinkSpeedA, LinkSpeedB These select the speeds of **Link0** and **Link1,2,3** respectively. Table 27.2 shows the possible combinations.

ClockIn A 5 MHz input clock for the transputer. The transputer synthesises its own high frequency clocks. **ClockIn** should have a stability over time and temperature of 200ppm. **ClockIn** edges should be monotonic within the range 0.8V to 2.0V with a rise/fall time of less than 8ns.

¹ Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Dual-In-Line Transputer Modules (TRAMs)* and *Module Motherboard Architecture* which are included later in this databook. The *Transputer Databook* may also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

LinkSpeedA	LinkSpeedB	Link0	Link1,2,3
0	0	10 Mbits/s	10 Mbits/s
0	1	10 Mbits/s	20 Mbits/s
1	0	20 Mbits/s	10 Mbits/s
1	1	20 Mbits/s	20 Mbits/s

Table 27.2 Link speed selection

Reset Resets the transputer, and other circuitry. **Reset** should be asserted for a minimum of 100ms. After **Reset** is deasserted a further 100ms should elapse before communication is attempted on any link. After this time, the transputer on the IMS B409 is ready to accept a boot packet on any of its links.

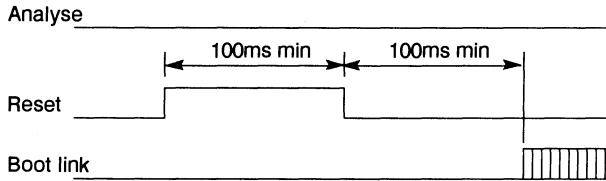


Figure 27.1 Reset timing

Analyse is used, in conjunction with **Reset**, to stop the transputer. It allows internal state to be examined so that the cause of an error may be determined. **Reset** and **Analyse** are used as shown in figure 27.2. A processor in analyse mode can be interrogated on any of its links.

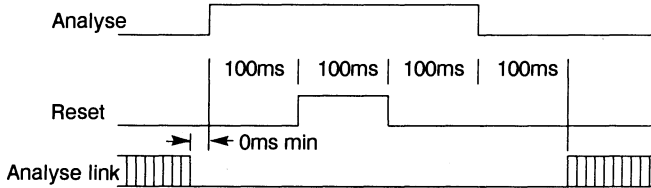


Figure 27.2 Analyse timing

notError An open collector output which is pulled low when the transputer asserts its Error pin. **notError** should be pulled high by a 10kΩ resistor to **VCC**. Up to 10 **notError** signals can be wired together. The combined error signal will be low when any of the contributing signals is low.

27.1.3 Pixel Bus connectors

The IMS B409 has three pixel data ports in addition to the usual TRAM signals. This enables the IMS B409 to connect via the INMOS pixel bus to one or more IMS B408s. The Pixel Bus uses 60-way IDC connectors and flat ribbon cable; the pinout of each port is defined in Table 27.3. The clock and control outputs are driven by high current buffers capable of driving into 100Ω loads.

Pin No.	In/Out	Pin	Pin	In/Out	Pin No.
1		GND	D0	in	2
3	in	D1	GND		4
5	in	D2	D3	in	6
7		GND	D4	in	8
9	in	D5	GND		10
11	in	D6	D7	in	12
13		GND	D8	in	14
15	in	D9	GND		16
17	in	D10	D11	in	18
19		GND	D12	in	20
21	in	D13	GND		22
23	in	D14	D15	in	24
25		GND	D16	in	26
27	in	D17	GND		28
29	in	D18	D19	in	30
31		GND	D20	in	32
33	in	D21	GND		34
35	in	D22	D23	in	36
37		GND	D24	in	38
39	in	D25	GND		40
41	in	D26	D27	in	42
43		GND	D28	in	44
45	in	D29	GND		46
47	in	D30	D31	in	48
49		GND	notSEQclk	out	52
51		GND	notRAMclk	out	50
53		GND	notFieldSync	out	54
55		GND	notEarlyBlank	out	56
57		GND	notEvenField	out	58
59		GND	SysReady		60

Table 27.3 Pixel Bus pin designations

D0 – 31 Pixel input data is latched on the falling edge of **notSEQclk**. The data bus is open collector and carries inverted data. This allows data from different serial port modules to be ORed on the Pixel Bus. Each data input is terminated with 330Ω to **VCC** and 470Ω to **GND**.

notSEQclk A continuous output clock for use as the system timing reference; it has a maximum frequency of 25 MHz. Data and control strobes transitions are synchronised to the falling edge of **notSEQclk**.

notRAMclk An output clock of the same frequency and phase as **notSEQclk** but gated so that it does not run during display blanking. It is used to clock pixel data from the IMS B408s onto the pixel bus so that no data is lost during blanking. In the off state it is high.

notEarlyBlank A time-advanced version of the display blanking signal. It provides early warning of when pixel data will be required and of when it should be turned off.

notFieldSync Output low during field flyback (vertical blanking) to indicate the start of a new field.

notEvenField For use in systems producing interlaced displays; e.g. TV standard displays. A low on this signal indicates that pixel data for the even field of the odd/even field pair making up an interlaced frame should be placed on the pixel bus. Changes state on the falling edge of **notFieldSync**.

SysReady Used as a synchronisation mechanism by multiple IMS B408 frame store modules. It is neither driven nor monitored by the IMS B409 but is common between the three pixel channel bus connectors to support the synchronisation mechanism.

27.1.4 The Pixel channels

The IMS B409 has three pixel channels: A, B and C. Each channel consists of a 4-1 byte multiplexer and an IMS G176 colour look-up table (CLUT). Input to a channel is through a pixel bus connector, output is from a set of RGB video outputs. There are two operating modes.

8 bits/pixel mode

Each channel accepts 32 bit pixel data from a pixel bus connector at 1/4 the pixel rate. This is multiplexed down to an 8 bit wide stream at pixel rate which is fed to the CLUT pixel data inputs. Thus, the pixel bus only runs at 1/4 the pixel rate which may be up to 64 MHz. Each channel can provide a separate display with up to 256 colours on each screen. It is not necessary to use all three channels. Since the three channels are synchronised it is possible to use each channel to generate one of the RGB primaries. Skew between any two pixel channels on the same IMS B409 is less than 5ns. Each channel would be connected by a separate pixel bus to one or more IMS B408s.

18 bits/pixel mode

In this mode the IMS B409 provides a single display of up to 262144 colours. This mode allows a full colour display to be produced by an IMS B409 with a single IMS B408. The pixel bus runs at the pixel rate which is therefore limited to 25 MHz. Each pixel requires a 32 bit word to be supplied to the channel A pixel bus input. The least significant byte is routed direct to the channel A CLUT, the second least significant byte is routed direct to the channel B CLUT, and the third least significant byte is routed direct to the channel C CLUT. Each CLUT is used to generate one of the RGB colour primaries. Thus, a single input word specifies directly the red, green and blue components of a pixel. Skew between any two pixel channels on the same IMS B409 is less than 5ns.

The colour look-up tables

Each of these devices combines a 256 word, 18 bit wide RAM and three 6 bit DACs. 8 bit data applied to the device's pixel inputs addresses a location in the RAM. 6 bits of the addressed data are applied to each of the DACs which generate the red, green, and blue (RGB) outputs. Thus, the device can display up to 256 colours, selectable from a palette of 262144. The RAM contents are writeable and readable by the IMS T222.

Video Outputs

Each pixel channel has a set of RGB outputs brought out on three SMB connectors. The outputs are current sources with 75 Ω termination and will drive 1V peak-peak into a 75 Ω load. The outputs are d.c. coupled: 0.3V is blanking level and 1.0V (on load) is peak white.

Sync is not composited with the video signals but is available from a separate sync output (also an SMB connector). The sync output will also drive into 75Ω and is d.c. coupled: 5V is the idle level, sync pulses are 0V.

27.1.5 Memory Map

The IMS B409 is able to access 4 kbytes of internal transputer memory. This is sufficient memory to contain the small amount of code and data required to set up the colour look-up tables and the VTG. The internal memory on the IMS T222 has a 50ns access cycle time; i.e. a single processor cycle. The IMS T222 has a 64 kbyte address space with addresses ranging from #8000 to #7FFF where # indicates a hexadecimal number.

	Byte Address
IMS T222 internal RAM	#8000-#8FFF

Table 27.4 IMS B409 memory location

Pixel Channel Mode select

The pixel channel mode is set by writing to the Pixel Channel Mode Select register. Writing 1 selects multiplexed (8 bits/pixel) mode: writing 0 selects non-multiplexed (18 bits/pixel) mode. The register location is given in table 27.5. This register is write only.

Register	Byte Address
Channel Mode Select	#B000

Table 27.5 Channel Mode Select register location

The video timing generator

The video timing generator is an NEC D7220. It is mapped into the IMS T222's address space as shown in Table 27.6. It is used only as a programmable timing generator and performs no drawing functions. Line frequency, field frequency and resolution can be programmed (horizontal resolution must be a multiple of 64 pixels) and displays may be either interlaced or non-interlaced.

Register	Byte address
Parameter FIFO (write only)	#A000
Status Register (read only)	#A000
Command FIFO (write only)	#A002
FIFO read (read only)	#A002

Table 27.6 NEC D7220 register locations

The Colour look-up tables

Ordinary accesses to the CLUT registers should be made at the addresses shown in table 27.7. These registers are mapped as the lower 8 bits of a 16 bit word addressed at that location. They can be written and read either as 16 bit words or as bytes addressed at the given locations. If written as 16 bit words, the upper 8 bits are ignored; if read as 16 bit words, the upper 8 bits are read undefined.

Register	Byte Address
Channel A Pixel Address (write mode)	#0000
Channel A Colour Value	#0400
Channel A Pixel Mask	#0800
Channel A Pixel Address (read mode)	#0C00
Channel B Pixel Address (write mode)	#1000
Channel B Colour Value	#1400
Channel B Pixel Mask	#1800
Channel B Pixel Address (read mode)	#1C00
Channel C Pixel Address (write mode)	#2000
Channel C Colour Value	#2400
Channel C Pixel Mask	#2800
Channel C Pixel Address (read mode)	#2C00

Table 27.7 IMS G176 registers

Each CLUT has a single Pixel Address register which is addressable at two locations. Writing a pixel address to the first, places the CLUT in colour value write mode. Writing a pixel address to the second places the CLUT in colour value read mode. Reading either location returns the same value.

Block moves to and from the colour value registers should be made to the regions defined in table 27.8. In this region, the colour value register appears as an 8 bit wide register at each byte address. Thus, byte arrays of colour values can be block copied to and from these areas. Correct results for block writes are not guaranteed for pixel clock speeds of less than 16 MHz. Correct results for block reads are not guaranteed for pixel clock speeds of less than 28 MHz.

Register	Byte Address
Channel A Colour Value	#4400-#47FF
Channel B Colour Value	#5400-#57FF
Channel C Colour Value	#6400-#67FF

Table 27.8 IMS G176 block move areas

27.1.6 Mechanical details

Figure 27.3 indicates the vertical dimensions of a single IMS B409 and Figure 27.4 shows the outline drawing of the IMS B409. Note that the component height includes the height taken up by a cable plugged into a pixel bus input. This means that the IMS B409 on a motherboard occupies more than one card slot in a 0.8in. pitch card cage.

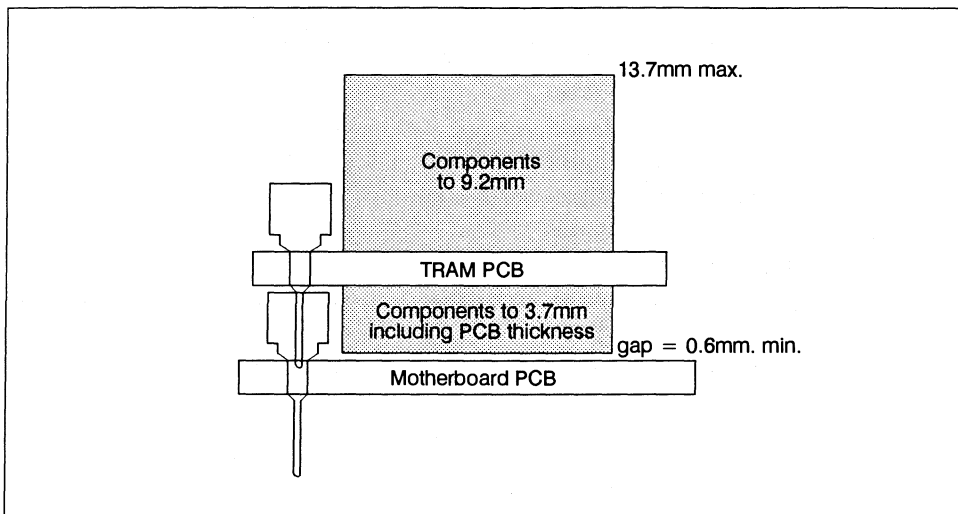


Figure 27.3 IMS B409 height specification

27.1.7 Installation

Since the IMS B409 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B409 may be supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B409 into the motherboard. Where the IMS B409 is being used with an INMOS motherboard, the yellow triangle marking pin 1 on the IMS B409 (see Figure 27.4) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B409, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B409 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

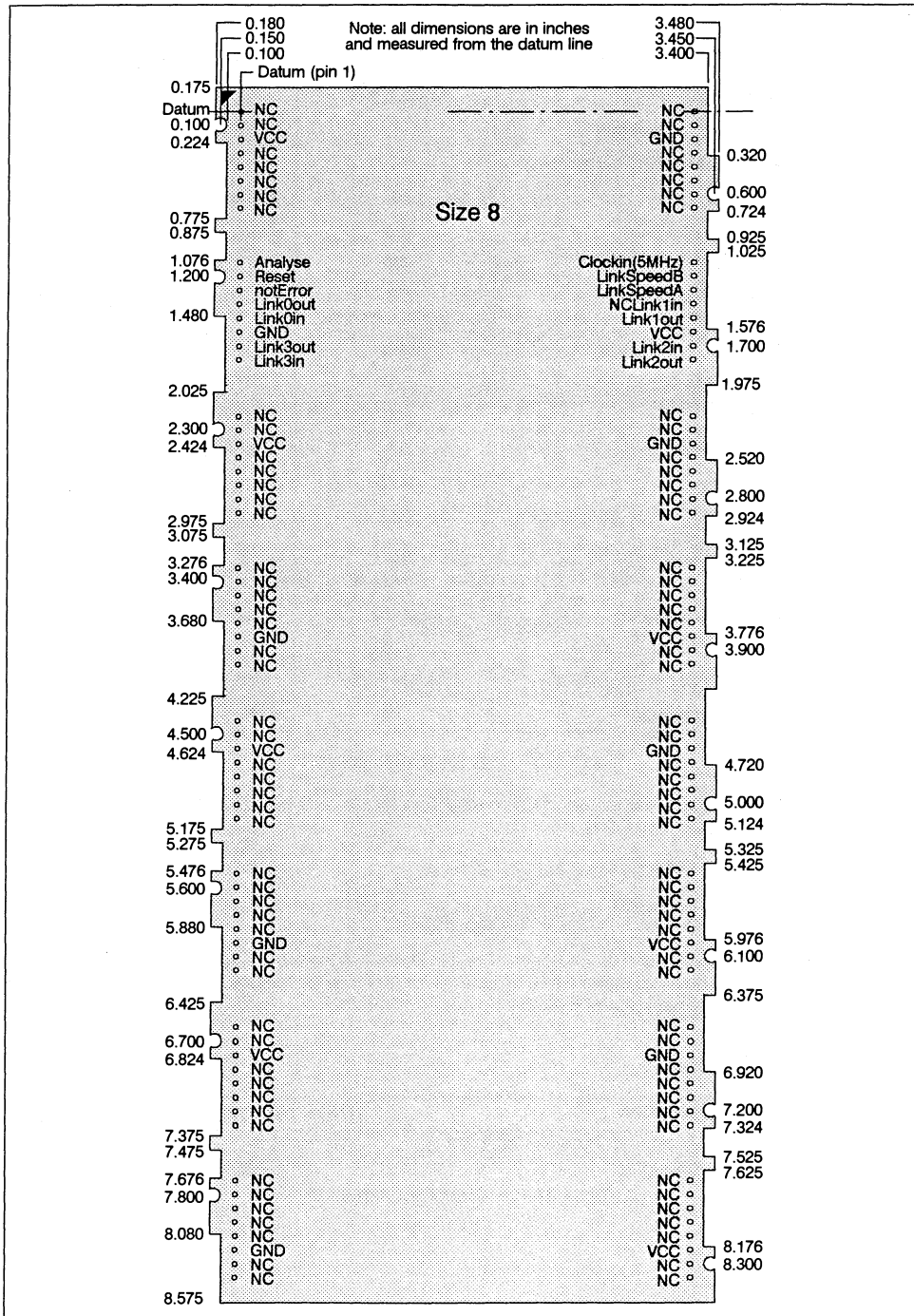


Figure 27.4 IMS B409 outline drawing (All dimensions in inches)

27.1.8 Specification

TRAM feature		Unit	Notes
Transputer type	IMS T222-20		
Number of transputers	1		
Number of INMOS serial links	4		
RAM size	4	kbyte	
Memory cycle time	50	ns	
Subsystem controller	No		
Peripheral circuitry	VTG 3 Display channels		
Parity	No		
Size (TRAM size)	8		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	8.75	inch	
Component height above PCB	12.8	mm	1
Component height below PCB	3.0	mm	2
Weight	185	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	3
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	18	W	4

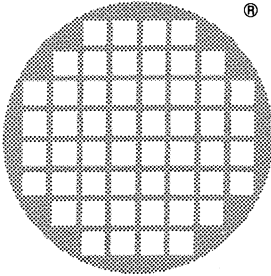
Table 27.9 IMS B409 specification

- 1 Since the IMS B409 makes use of IDC connectors for the pixel bus, this dimension is larger than is normally stated for TRAMs.
- 2 This dimension includes the thickness of the PCB.
- 3 The figure quoted refers to the ambient air temperature.
- 4 The power consumption is the worst case value obtained when a sample of IMS B409 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

27.1.9 Ordering Information

Description	Order Number
IMS B409 TRAM with IMS T222-20	IMS B409-1

Table 27.10 Ordering information

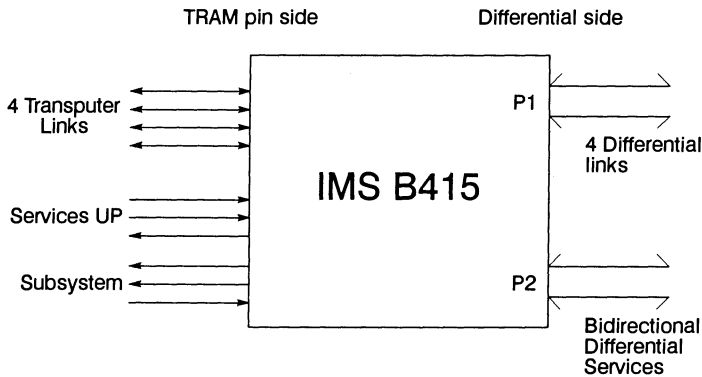


inmos[®]

IMS B415

Differential link buffer TRAM Size1

Engineering Data



FEATURES

- Buffers all TRAM signals to RS422 compatible differential drive
- Handles 4 links, reset and subsystem services signals
- Capable of 20 Mbit/s link operation
- Links go quiet when disconnected
- Designed for 100 ohm twisted pair cable
- $\pm 7V$ common-mode noise rejection
- Size 1 TRAM
- Designed to a published specification (*INMOS Technical Note 29*)

GENERAL DESCRIPTION

The IMS B415 Differential interface buffer TRAM allows connections between transputer systems which are not in the same electrical environment. No common ground connection is required, reducing earthing problems. With cable lengths up to 10m, 20Mbit/s link speed is possible. Longer cables up to 100m support lower link speeds.

28.1 Description

The IMS B415 is an INMOS TRANputer Module (TRAM) incorporating differential driver/receivers which allow INMOS serial links and control signals to be connected reliably between different pieces of equipment.

TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort. TRAMs may be plugged into motherboards, which provide the necessary electrical signals, mechanical support and usually, an interface to a host machine. Various motherboards are now available from INMOS and from third-party vendors for most of the common computing platforms.

Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in *Module Motherboard Architecture* and *Dual-In-Line Transputer Modules (TRAMs)* which are included later in this databook.

If the user intends to design a custom motherboard, then *The Transputer Databook* will also be required. This is available as a separate publication from INMOS (72 TRN 203 01).

28.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
Vcc, GND		Power supply and return	3, 14
NC1	in	Not connected	8
Reset	in	'Up' reset	10
Analyse	in	'Up' analyse	9
notError	out	'Up' error (active low))	11
Links			
LinkIn0-3	in	INMOS serial link inputs to differential buffers	13, 5, 2, 16
LinkOut0-3	out	INMOS serial link outputs from differential buffers	12, 4, 1, 15
NC2, NC3	in	Not connected	6, 7
Subsystem Services			
SubsystemReset	out	Subsystem Reset	1b
SubsystemAnalyse	out	Subsystem Analyse	1c
notSubsystemError	in	Subsystem Error Indicator	1a

Table 28.1 IMS B415 TRAM Pin designations

Notes

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in figure 28.3.

28.3 Introduction to the IMS B415

The IMS B415 is a TRAM designed to allow transputer links and system services to be connected between pieces of equipment which do not share a common power supply. It uses differential drivers and receivers which comply with the voltage levels used in IEC specification RS422. Link Speeds of up to 20Mbits/s are possible (with the correct cable) and common mode voltages of up to 7V are accommodated. The IMS B415 provides a simple and reliable method for connecting various transputer-based systems together, and is ideal when communicating between a development system board (for instance an IMS B008 in a PC or an IMS B014 in a workstation) and a target system. When the target system is some distance from the development system or where there is no mains earth wire joining the two systems it is particularly important to use some kind of isolated or differential signalling, such as the IMS B415 provides.

28.4 Principles of Operation

The IMS B415 provides differential buffers for four transputer links and for associated system services signals (Reset, Analyse and Error). The buffers used provide the very low signal skew necessary for reliable link operation at 20Mbits/s.

With suitable cables, two coupled IMS B415 TRAMs provide buffered link connections between the two TRAM slots onto which they are fitted. If 'subsystem' pins (three-pin double-ended strip) are fitted to one of the IMS B415s then the services signals received by the other TRAM will be propagated out of the 'subsystem' services port. Usually one IMS B415 would be fitted to a slot on a development system motherboard (IMS B008, B014, B015), while the other would be fitted to a target system motherboard. The second IMS B415 would be installed in a slot with 'subsystem' capability and would have the 'subsystem' pin strip installed.

The IMS B415 contains only differential drivers/receivers and associated line conditioning components. This means that the correct link behaviour is maintained and no software changes are required¹. The IMS B415's schematic is provided for users to understand the operation of the TRAM.

28.5 Differential Connectors

Two 20-pin connectors, designated 'P1' and 'P2' carry the differential signals from the top of the TRAM. These are standard 0.025 inch square post pin headers, with two rows of ten pins on 0.1 inch centers. The contacts are gold plated to commercial class II. P1 carries the link signals while P2 carries the system control signals. Pin designations are shown in table 28.2.

P1		P2	
Pin Number	Signal Name	Pin Number	Signal Name
1	Link0Out-	1	DownReset-
2	Link0out +	2	DownReset +
3	Link0In-	3	DownError-
4	Link0In +	4	DownError +
5	Link1Out-	5	DownAnalyse-
6	Link1out +	6	DownAnalyse +
7	Link1In-	7	UpReset-
8	Link1In +	8	UpReset +
9	Link2Out-	9	UpError-
10	Link2out +	10	UpError +
11	Link2In-	11	UpAnalyse-
12	Link2In +	12	UpAnalyse +
13	Link3Out-	13	Not Connected
14	Link3out +	14	Not Connected
15	Link3In-	15	Not Connected
16	Link3In +	16	Not Connected
17	Cut short for polarization	17	Not Connected
18	Not Connected	18	Not Connected
19	Cut short for polarization	19	Cut short for polarization
20	Not Connected	20	Cut short for polarization

Table 28.2 P1 and P2 pin designations

¹ Buffering systems which decode the link packets and forward them using a different coding technique would require software modifications because more than one byte may be in transit in the link at any one time. The IMS B415 and cable will introduce a delay into the link signals which will reduce the link data rate by an amount dependent upon the transputers used.

Many different mating connectors can be used, including standard 'IDC-type' ribbon cable connectors. However, in order to reduce the distance by which the connectors overhang the edge of the TRAM, the cables shipped with IMS B415-1 are the DuPont shell and crimp type. Shells are DuPont part number 47564-002. Crimps are part number 65043-027.

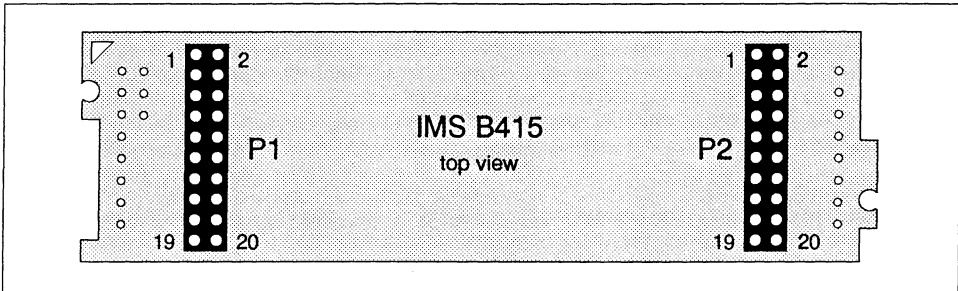


Figure 28.1 Connector positions and pin numbers

28.6 Cables

The IMS B415 is designed to operate with cables which have a characteristic impedance of 100Ω. Long cables will degrade the signal and cables longer than 10m are unlikely to give reliable operation at 20Mbits/s. INMOS recommends that cables are of the twisted-pair variety and do not have an earthed screen. However, many different cable configurations are possible at the discretion of the user. Note that FCC electromagnetic emissions requirements will not be met unless screened cable is used in an unshielded enclosure. See references [1] and [2] for a detailed treatment of the issues involved. For normal operation between two IMS B415 TRAMs, cables wired as detailed in table 28.3 should be used. Note that adjacent pins must always be wired as a twisted-pair. The cable used in IMS B415-1 is SpectraStrip, manufactured by Amphenol, part number 1352802-316.

P1		P2	
First IMS B415	Second IMS B415	First IMS B415	Second IMS B415
1	3	1	7
2	4	2	8
3	1	3	9
4	2	4	10
5	7	5	11
6	8	6	12
7	5	7	1
8	6	8	2
9	11	9	3
10	12	10	4
11	9	11	5
12	10	12	6
13	15		
14	16		
15	13		
16	14		

Table 28.3 Cable wiring for P1 and P2

28.7 Mechanical details

Figure 28.2 indicates the vertical dimensions of a single IMS B415 TRAM, and Figure 28.3 shows the outline drawing of the IMS B415.

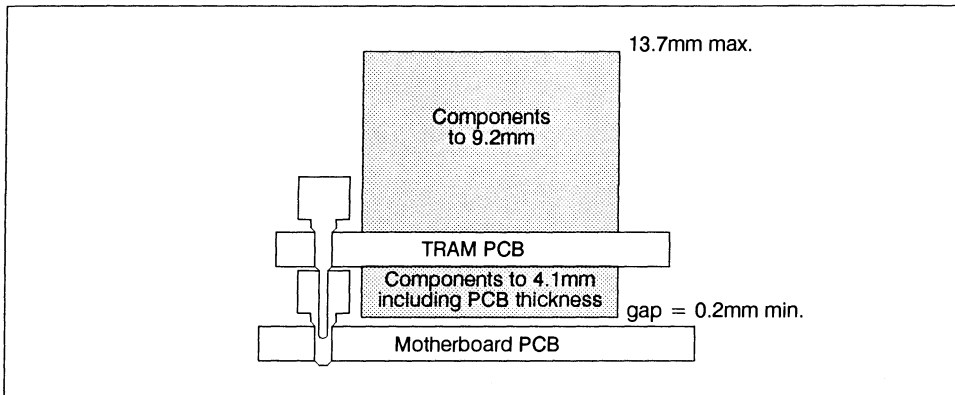


Figure 28.2 IMS B415 height specification

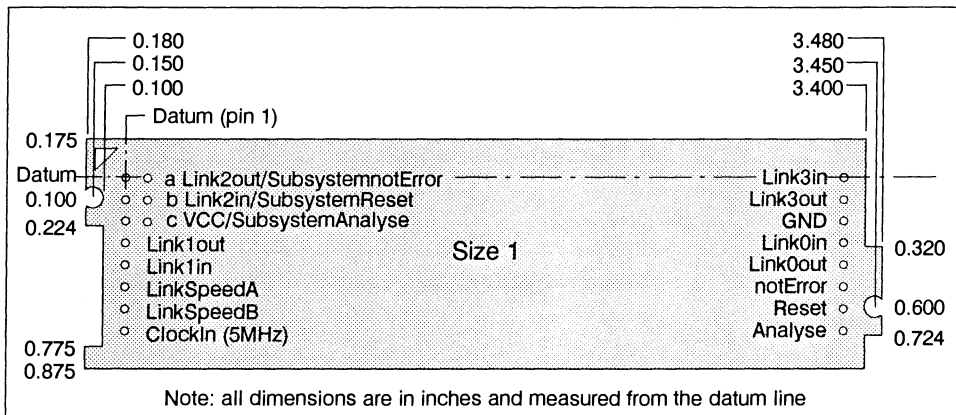


Figure 28.3 PCB profile drawing and pinout

28.8 Installation

Since the IMS B415 contains ESD sensitive components, all normal precautions to prevent static damage should be taken.

The IMS B415 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

Plug the IMS B415 carefully into the motherboard. Where the IMS B415 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B415 (see figure 28.3) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot. If it is envisaged that the assembly is likely to be subjected to any vibrations, it is recommended that the TRAM is secured to the motherboard using nylon M3 nuts and bolts. The bolts should be inserted through the fixing holes

on the motherboard, and through the castlations on two edges of the TRAM. A number of these nuts and bolts are supplied with each of the INMOS motherboards.

Should it be necessary to unplug the IMS B415, it is advised that, having removed any retaining nuts and bolts, it is gently levered out while keeping it as flat as possible. As soon as the IMS B415 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

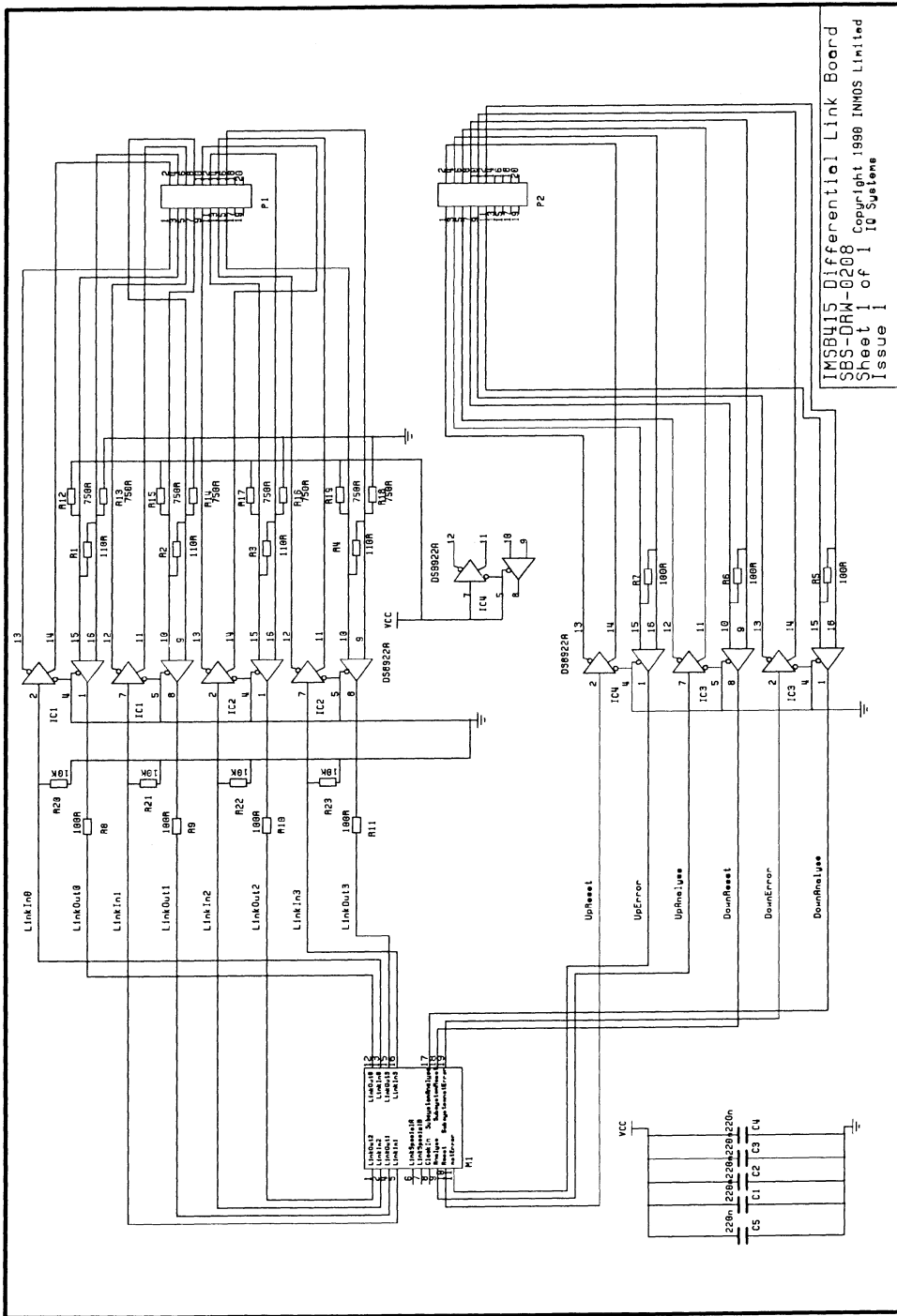
28.9 Specification

TRAM feature	IMS B415-0	IMS B415-1	Unit	Notes
Number of INMOS serial links	4	4		
Subsystem controller	Yes	Yes		
Peripheral circuitry	None	None		
Size (TRAM size)	1	1		
Length	3.66	3.66	inch	
Pitch between pins	3.30	3.30	inch	
Width	1.05	1.05	inch	
Component height above PCB	9.2	9.2	mm	
Component height below PCB	3.7	3.7	mm	1
Weight	20	20	g	
Storage temperature	0-70	0-70	°C	
Operating temperature	0-50	0-50	°C	2
Power supply voltage (Vcc)	4.75-5.25	4.75-5.25	Volt	
Power consumption	2	2	W	3
Common mode noise rejection	7	7	v	
Propagation Delay	10	25	ns	
Cables supplied	No	Yes		
Cable Length	NA	1	m	

Table 28.4 IMS B415 specification

Notes

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B415 TRAMs were tested (With differential outputs terminated in 100 ohms) at a supply voltage (VCC) of 5.25V.



IMSB415 Differential Link Board
 SBS-DRW-0208 Copyright 1998 INMOS Limited
 Sheet 1 of 1 I/O Systems
 Issue 1

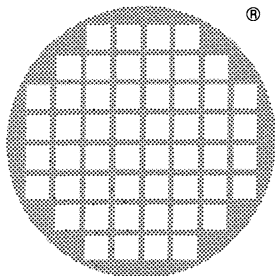
28.10 References

- 1 *Microcomputer Interfacing* , Harold S. Stone, Addison-Wesley, 1982.
- 2 *MECL System Design Handbook* , William R. Blood, Jr., Motorola Inc., 1983.

28.11 Ordering Information

Description	Order Number
IMS B415 TRAM	IMS B415-0
Pair of IMS B415 TRAMs with cables	IMS B415-1

Table 28.5 Ordering information



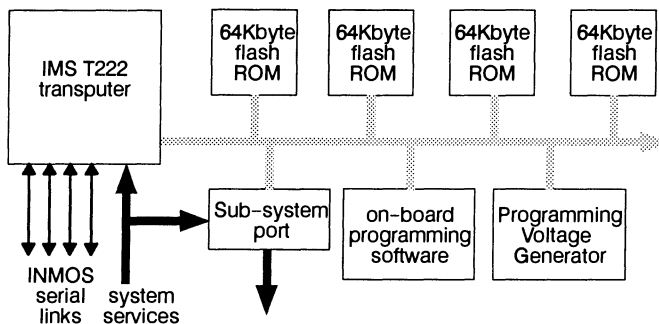
inmos[®]

IMS B418

Flash ROM TRAM

Size 2

Engineering Data



FEATURES

- 256 Kbytes non-volatile memory (Flash ROM)
- 10 000 program/erase cycles
- Ideal for booting embedded transputer systems
- In-system reprogrammability
- ROM contents user-programmed through INMOS link
- Size 2 TRAM
- Sub-system port for resetting transputer networks
- Can be used as non-volatile backup memory
- Includes on-board programming software
- Designed to a published specification (*INMOS Technical Note 29*)

GENERAL DESCRIPTION

The IMS B418 is a TRAM designed primarily for configuring and boot-strapping transputer networks in embedded systems. It contains 256 Kbytes of non-volatile memory implemented with *flash ROM* devices (the *flash ROM* is an EPROM-like device with bulk electrical erasability, rather than UV erase).

After reset, the IMS B418 outputs a program stored in the ROM from one of its INMOS serial links. An on-board programming voltage generator, and programming software, allows the ROM contents to be programmed without removing the ROM devices from the board and without removing the IMS B418 from an assembled system. Programming is through a simple protocol on one of the INMOS serial links. Safeguards are provided against accidental erasure/programming. The ROM devices can be reprogrammed at least 10 000 times.

The IMS B418 can also be used as a non-volatile backup memory in any microprocessor system; all that is required is an INMOS link adaptor to interface the IMS B418 to the microprocessor.

29.1 Description

29.1.1 Booting transputer networks with the IMS B418

Transputers and transputer networks can be boot-strapped by inputting the program as a stream of bytes on a single transputer link. The IMS B418 provides a means of storing the program for a network of transputers in an embedded system. The phrase 'target system' is used to describe the transputer or network of transputers which is bootstrapped by the IMS B418.

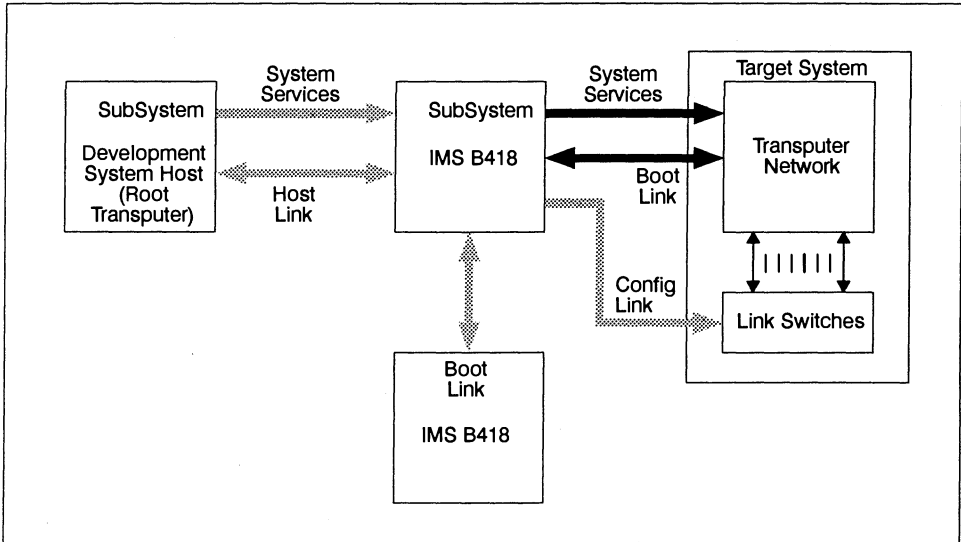


Figure 29.1 The IMS B418 in use

Figure 29.1 shows how the IMS B418 is used. The required connections are shown as solid lines, optional connections by shaded lines.

In a completed system incorporating an IMS B418, reset signals from the IMS B418 (*subsystem services*) are used to reset the target system. The IMS B418 has a power-on reset circuit and so provides power-on reset and automatic start-up of the target system. The IMS B418 will also have one or two link connections to the target system. One of these carries program code from the IMS B418 to the target system; the other, if present, can be used to configure a set of link switches prior to boot-strapping the target.

During software development for the target system, the IMS B418 can be incorporated into the system as it would be in the finished product. The development host can be connected to the IMS B418, as shown in figure 29.1. Transparency modes give the host the same view of the target system as the IMS B418 has.

In a completed system it may be useful to allow outside access to the services signals going into the IMS B418 and to one of the IMS B418 links (as in the the host connections shown in figure 29.1). This access would allow field updates to the system software.

If the application code is larger than the 256Kbytes provided by a single IMS B418, a second IMS B418 can be added as shown in figure 29.1. Similarly, a third IMS B418 can be connected to the second, and so on, up to a maximum of four IMS B418s. Such a chain or cascade of boards appears the same to the programmer as a single board with a larger amount of ROM.

29.1.2 Flash ROMs

The IMS B418 uses *Flash ROM* devices. These are memory devices which, like EPROMs, retain their contents when power is removed. Unlike EPROMs, the entire memory contents can be erased by a command

sequence from a microprocessor. The device may then be reprogrammed. The devices used on the IMS B418 can be erased and reprogrammed up to 10 000 times.

Programming a flash ROM device requires the programming voltage to be established, and requires a sequence of command bytes to be sent to the device. It is very unlikely therefore that a flash ROM device on the IMS B418 could have its contents modified accidentally. To provide extra protection against accidental erasure or programming, the programming voltage can be disconnected from the flash ROMs by removing J8 and J9. J9 is normally fitted to enable programming and erase operations by users. J8 is used to enable factory programming of the IMS B418 resident software and should not normally be fitted.

Programming a flash ROM can change any bit from a 1 to a 0: to change a bit from 0 to 1 requires the device to be erased. In addition, to prevent over-programming, the programming software on the IMS B418 prevents bytes which have already been written (bytes which do not contain #FF¹) from being re-written. It does not erase devices which are already erased.

Note that, in use, the details of the programming mechanism are hidden. The user sends bytes to be programmed to the IMS B418 using a simple protocol.

29.1.3 The IMS B418 In the development environment

The following points should be noted:

- 1 Bootable programs produced using the INMOS software development tools can be copied directly to an IMS B418, which can then bootstrap a target system.
- 2 The IMS B418 is compatible with the debugging tools supplied with the INMOS software development tools.

The IMS B418 has a number of different operating modes which are selected by jumpers depending on the requirements of the development environment. Typically, initial development can be done with the IMS B418 in *transparent mode*. The final stages of development can be accomplished using a combination of *auto-program mode* and *bootstrap mode*.

More detailed information on the IMS B418 operating modes is given in [1].

Bootstrap Mode

Bootstrap mode is used when the IMS B418 is installed in a completed product when performing its intended function of bootstrapping a transputer network. This mode may also be used during software development for the target system. Once the IMS B418 has boot-strapped the target network, it enters command mode. The target system can then use read and write block commands (see section 29.1.4) to store data in the IMS B418. During software development, it may be necessary to debug this and other parts of the application.

The IMS B418 propagates the reset and analyse signals applied to it, to the target system connected to its subsystem port. Thus, analysing an IMS B418 analyses the target network. It also causes the IMS B418 to behave transparently between the link it used to bootstrap the target network, and the first link on which it receives a byte after being analysed. A development host can therefore analyse and debug a target system which has been bootstrapped by an IMS B418, by running development system tools (such as `idebug`) in the normal way.

Transparent mode

Transparent mode can be useful during the early stages of software development. It is desirable during development to have the IMS B418 present in its intended place in the target system, so that the development hardware is identical to the product system. In transparent mode, the IMS B418 behaves completely transparently between a development host and a target system; it performs no bootstrap operation. The target software can then be developed as if the IMS B418 were not there. The development system tools such as `idebug` can be used as normal since, when the target system is analysed or reset, the IMS B418 behaves transparently.

1. Throughout this manual, the notation # is used to signify a hexadecimal number.

Auto-program mode

Auto-program mode provides an easy way of placing the bootable program and network configuration code for the target system in the IMS B418. The bootable program must be configured using the `iconf`¹ or `icollect`² development tools.

29.1.4 Firmware

The IMS B418 has some resident software which programs data into the flash ROMs, bootstraps the target system and implements the various transparency and debugging modes. It interprets a command protocol which is listed below. This firmware is stored in a ROM device separate from those used to store user-defined bootstrap and configuration code. This software cannot be erased or destroyed except by placing the IMS B418 in *boot from link* mode (by fitting J7) and loading a program which erases it. This is used for factory programming of the IMS B418 firmware.

DO NOT UNDER ANY CIRCUMSTANCES FIT J7

Note that the IMS B418 does not execute any part of the user defined bootstrap or configuration code.

The firmware is used by sending commands to the IMS B418, when in bootstrap mode, on any link. When a command is received on a link, it performs an appropriate action and makes a response to the same link.

The command protocol allows configuration and bootstrap sequences to be written and also allows specific blocks to be written and read. The read and write block commands allow the block contents of generically programmed boards to be patched with specific data and allow the IMS B418 to be used for data storage.

Programming a chain of IMS B418s is exactly the same as a single IMS B418, the user simply seeing a larger address space.

Summary of Programming Protocol

Commands:

Write boot packet	BYTE 'b'; INT32 count; [count]BYTE array
Write config packet	BYTE 'c'; INT32 count; [count]BYTE array
Erase device	BYTE 'e'; INT16 device
Read block	BYTE 'r'; INT32 address; INT32 count
Write block	BYTE 'w'; INT32 address; INT32 count; [count]BYTE array
Query status	BYTE 's'
Query firmware version	BYTE 'v'; INT16 n
Reboot target	BYTE 'x'

29.1.5 Programming Voltage Generator

The IMS B418 incorporates an on-board programming voltage generator for the flash ROM devices. The programming voltage is normally off but is turned on by the on-board programming software when a program or erase command is received. The power-on reset circuit on the IMS B418 disables the programming voltage in the event of a power failure and when power is applied to the IMS B418. Thus, the programming voltage can only be turned on by the programming software, which ensures the safety of the programmed ROM contents.

1. LFF toolsets

2. TCOFF toolsets

29.1.6 Power-on reset / Power-fail monitor

The IMS B418 incorporates a power-fail monitor and power-on reset generator. If the Vcc supply falls below 4.6V, the power-fail monitor will turn off the programming voltage generator and reset the IMS T222 on the IMS B418. It also de-asserts the **reset** and **analyse** signals on the IMS B418's *sub-system* pins. When power returns the IMS B418 is held reset for a few ms. It then resets and reboots the target system.

29.1.7 Sub-system Pins

The IMS B418 is designed to configure and bootstrap a network of TRAMs or transputers, and must be able to reset them before it can do so. To allow this, it incorporates a set of pins which allow it to control the **reset** and **analyse** signals of other TRAMs and to monitor their combined **error** signals. These are located in the corner of the IMS B418 marked with a yellow triangle. They are designed to mate with a corresponding set of pins on a module mother-board.

29.1.8 Link select jumpers

The firmware requires the user to select one or two of the IMS B418's transputer links for various functions. The selection is made by installing the appropriate combination of two pairs of jumpers: J1 and J2 form one pair of link select jumpers, J3 and J4 form the other pair. Table 29.1 shows which jumpers should be inserted to select a particular transputer link with each pair of jumpers. For example, to select link 2 as the J1/J2 link, fit J2 and remove J1. The function of these links depends on the operating mode of the IMS B418; more information is given in [1]. The J1/J2 link is normally connected to the target system and the J3/J4 link is either connected to a development host, or is the network configuration link.

Link	J1/J2 Link	J2/J3 Link
0	- -	- -
1	J1 -	J3 -
2	- J2	- J4
3	J1 J2	J3 J4

Table 29.1 Link selection jumpers

When installed in a target system (outside the development environment), these jumpers do not need to be changed.

29.1.9 Mode select jumpers

The behaviour of the IMS B418 depends on the state of the *mode select* jumpers J5 and J6 (see table 29.2). These two jumpers are read at power-on or reset. To change the operating mode, you will need to set these jumpers appropriately, then reset or power-cycle the IMS B418.

J5	J6	Mode
remove	remove	bootstrap
fit	remove	transparent
remove	fit	undefined
fit	fit	auto program

Table 29.2 Mode selection jumpers

29.2 Specifications

TRAM feature		Unit	Notes
Flash ROM	256	kbytes	1
TRAM size	2		
Length	3.66	inch	
Width	2.15	inch	
Pitch between pins	3.30	inch	
Component height above PCB	9.2	mm	
Component height below PCB	3.5	mm	2
Weight	40	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	
Power supply voltage (Vcc)	4.75-5.25	Volt	
Power consumption (Max)	4	W	3
Power consumption (Typical)	2.2	W	3

Table 29.3 IMS B418 specification

Notes

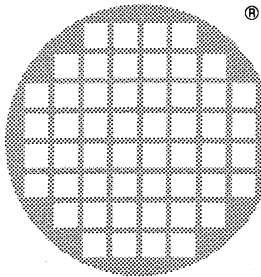
- 1 262128 bytes are available to the user: the on-board programming software used on the IMS B418 is stored in a separate device.
- 2 This dimension includes the thickness of the PCB.
- 3 Typical supply current is 400mA but, during program and erase operations, peaks of 800mA may be drawn.

29.3 Reference

- 1 IMS B418 User Guide, Inmos Limited, 1989

29.4 Ordering Information

Description	Order Number
IMS B418 Flash-ROM TRAM	IMS B418-10



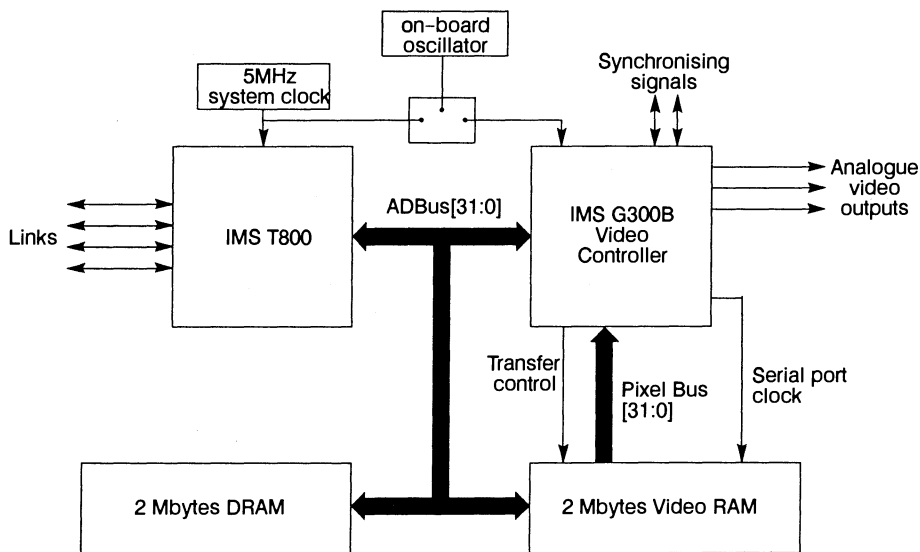
inmos[®]

IMS B419

Integrated graphics TRAM

Size 6

Engineering Data



FEATURES

- IMS T800-20 32 bit Transputer
- IMS G300B Colour Video Controller
- 2 Mbytes of four cycle DRAM
- 2 Mbytes of four cycle VRAM
- Huge variety of software selectable screen formats
- Pixel rates 25 to 100 MHz @ 8 bit/pixel
- Communicates via 4 INMOS serial links (Selectable between 10 or 20 Mbits)
- Size 6 TRAM
- Designed to a published specification (*INMOS technical Note 29*)
- Supplied with IMS F003 2D graphics library

GENERAL DESCRIPTION

The IMS B419 incorporates the IMS G300B Colour Video Controller (CVC) with the IMS T800 32 bit Floating Point Transputer to form a high performance graphics system. Two Mbytes of four cycle DRAM provides a general purpose store sufficient to run large applications such as windowing environments. Two Mbytes of Video RAM provide arbitrary screen resolutions up to a maximum of 1280 x 1024 8 bit/pixel with unrestricted screen formats at resolutions below this.

30.1 Description

30.1.1 Introduction

The IMS B419 is one of a range of INMOS TRANputer Modules (TRAMs). TRAMs are board level transputers with a simple, standardised interface. They integrate processor, memory and peripheral functions allowing powerful, flexible, transputer based systems to be produced with the minimum of design effort¹.

The IMS B419 implements a complete high performance graphics subsystem. The frame store consists of 2 Mbytes of dual ported Video RAM which supports displays of arbitrary resolution at 8 bit/pixel. The resolution of the system is programmable and is only limited by the CVCs maximum dot rate and the access time of the serial port on the VRAM. The IMS B419 supports a dot rate up to 100 MHz, the speed of the CVC. The CVC is configured by an IMS T800 which is provided with 2 Mbytes of 200ns cycle DRAM. This store is available for screen manipulation workspace and general program memory. The processor can be used to implement graphic primitives directly or as an intelligent channel, receiving image data from other transputers via its four bidirectional links at data rates of up to 6.8 Mbytes/sec. This makes the IMS B419 useful for applications as diverse as an add-on accelerator for a PC or a Macintosh, as part of an embedded system in industrial control, or as a graphics output for a 3D graphical supercomputer.

30.1.2 Screen sizes

Screen sizes are set by writing to a few registers in the G300B CVC, and can be chosen to suit the application. Suppose, for instance, an 8.5 x 11 sheet of paper (in landscape), represented by a screen with 100 pixels per inch. This would need an 1100 x 850 display, a format not normally available from a hardware solution. The G300B gives a line width in multiples of 4 pixels, which makes it simple to produce this screen. As well as producing special screens such as 11 x 8.5, many of the standard screens can also be produced; indeed the user can switch between screen formats, the display clock frequency, and even the source of the input clock, all by simply changing the G300B registers and other registers on the board by software.

Some examples of possible screen sizes are given in Table 30.1. All the screens in the table are for 8 bits per pixel.

Screen Size	Pixels	Aspect Ratio	Interface
CGA	320 x 240	1.333	no
EGA	640 x 350	1.829	no
VGA	640 x 480	1.333	no
Enh VGA	800 x 600	1.333	no
Ext VGA	1024 x 768	1.333	no
11 x 8.5	1100 x 850	1.294	no
11 x 8.5	1164 x 900	1.293	no
	1024 x 1024	1.0	no
	1280 x 1024	1.25	no
A5	1216 x 860	1.414	no

Table 30.1 A selection of possible screen sizes

1. Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in technical notes *Dual-In-Line Transputer Modules (TRAMs)* and *Module Motherboard Architecture* which are included later in this databook. The *Transputer Databook* may also be required. This is available as a separate publication from INMOS (72 TRN 203 01)..

30.1.3 SubSystem signals

The user may require the G300B Graphics TRAM to control a network of transputers and/or other TRAMs. A set of control signals are provided which enables the master to control these slaves or subsystems. The SubSystem port consists of three signals: **SubSystemReset** and **SubSystemAnalyse**, which enables the master to reset and analyse its subsystem; and **SubSystemnotError**, which is used to monitor the error flag in the subsystem.

These signals are accessible to the processor as a set of memory-mapped registers.

Register	Hardware byte address	Asserted state
SubSystemReset (Wr only)	#00000000	1
SubSystemAnalyse (Wr only)	#00000004	1
SubSystemError (Rd only)	#00000000	1

Table 30.2

30.1.4 CVC reset register

The IMS G300B must be reset before it can be programmed. It allows users to reset the IMS G300B CVC from software running on the IMS T800. To reset the CVC, the CVC reset register must be written with 1 for a minimum of 10 μ s, then cleared. The CVC reset register is located at #000000F0.

CVC reset register	IMS G300 state
1	Reset
0	Enabled

Table 30.3 CVC reset register.

30.1.5 Clock Select Register

The clock select register allows users to choose a pixel dot rate which may not be possible using the 5 MHz TRAM clock and PLL multiplication factors, or which is below the range of the PLL. The clock select register is located at #000000F4. On power up, or on a system reset the clock select register defaults to '0'.

PLL clock select register	Clock source
0	5MHz TRAM clock
1	Crystal Oscillator

Table 30.4 PLL mode clock source selection

30.1.6 Memory Map

The memory space on the board may be divided up into two non-contiguous areas, screenspace and workspace, so that operating systems which use automatic workspace sizing will not trespass on the screen space. Alternatively, if the drawing program requires over 2 Mbytes and not much screen space is required, the memory can be arranged so that the VRAM is contiguous with the workspace RAM. Figure 30.1 shows how the memory is mapped into the address space of the IMS T800.

Jumper Fitted	VRAM start address
JP4	#80200000
JP5	#C0000000

Table 30.5 VRAM Start Address selection

NOTE: JP4 and JP5 **must not** be fitted at the same time, or damage may result.

The selection is made by fitting one or other of a pair of jumpers.

Figure 30.1 shows how the memory is mapped into the address space of the IMS T800 (the “#” sign indicates a hexadecimal number).

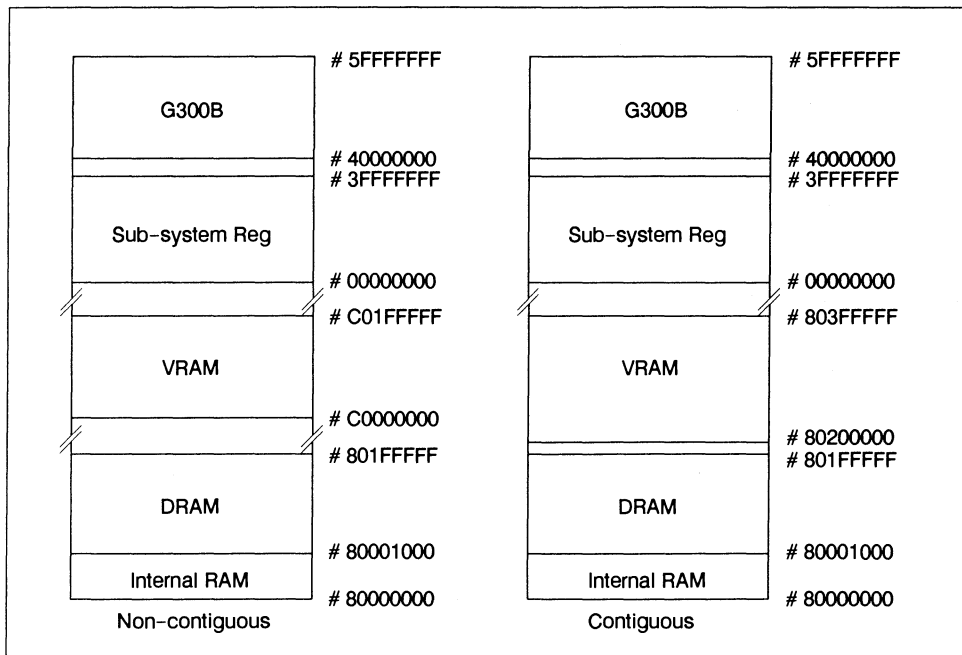


Figure 30.1 Non-contiguous and contiguous address maps

Information on the G300B CVCs registers and locations can be found in [2].

30.1.7 Pixel clock selection

The IMS G300B requires a clock to control the movement of pixel data, and generate timing signals. It has a phase-locked loop (PLL) which can generate the high frequency pixel clock from a low frequency input clock. The PLL can generate frequencies from 25MHz upwards. On the IMS B419-4, the pixel clock must be in the range 25MHz-100MHz. The IMS B419, provides a choice of clocking schemes: choosing a clocking scheme must be done partly when the IMS B419 is installed in a system, and partly by the user software whenever the system starts up. At installation time, the clocking scheme you choose to use determines whether you need to fit a crystal oscillator module to the IMS B419. At system start-up, your software may have to program a multiplication factor for the PLL and select a clock source for the PLL. The clocking schemes are all described below. Factors influencing the choice of clocking scheme are: whether the required clock is within the range achievable with the PLL; and, if so, if it is a multiple of 5MHz.

5MHz TRAM clock and PLL The primary clocking system utilises the IMS G300's on-chip phase-locked loop (PLL) to multiply the 5MHz TRAM clock to the video data rate. The multiplication factor must be an integer value between 5 and 20 to produce a video data rate in the range of 25-100 MHz. The clock select register must be written with 0 to select this mode. Bit 5 of the IMS G300B boot location must be set to enable the PLL.

Crystal oscillator and PLL The second method uses the on board crystal oscillator to drive the PLL clock input. This method is used when the required video data rate is not a multiple of 5MHz, but

is within the range of the PLL. The clock select register must be written with 1. Bit 5 of the IMS G300B boot location must be set to enable the PLL.

Any oscillator frequency in the range of 5.0-9.0 MHz may be used. The crystal oscillator module is socketed to make replacement easy. The oscillator module must be as specified in [1]. The resulting pixel clock should be in the range 25MHz-100MHz and the clock multiplication factor must not be less than 5.

Although all possible multiplication factors will work with all permissible input frequencies; it is recommended that, for any particular output frequency, the minimum suitable multiplication factor should be used. For example, to generate an 80MHz pixel clock you could multiply the 5MHz TRAM clock by 16, but fitting an 8MHz crystal oscillator and setting the PLL multiplication factor to 10 will produce a more stable pixel clock.

× 1 Mode The third method is to operate the IMS G300 in × 1 clock mode from the onboard crystal oscillator. This method is not recommended. Bit 5 of the IMS G300B boot location must be written with 0 to disable the PLL. The clock select register must be written with 1. The clock signal in × 1 mode must be low for a minimum of 6ns, the maximum pixel clock frequency in this mode is approximately 80MHz. Note that in this mode, you must still write a PLL multiplication factor to the bootstrap location, even though it is not used. The value 5 is suggested.

30.1.8 Jumper selection

Five jumper links are used to select the IMS G300B clock source and to configure the memory map of the IMS B419. Jumpers are labelled JPx where a jumper is either installed or absent between two pin posts.

Jumper	Function
JP1	Always remove on IMS B419-4
JP2	Do not fit
JP3	Always fit
JP4	Select contiguous VRAM
JP5	Select non-contiguous VRAM

Table 30.6

30.1.9 Video and sync outputs

The G300B CVC can be programmed to generate timing which complies with both the RS170a and EIA-343 video standard. The outputs are designed to drive a 75R line directly. The RGB analogue outputs and synchronising signals are on five SMB connectors as shown below. If the display monitor accepts composite sync on one of its video inputs the sync outputs may be left unconnected. SMB identification from top to bottom of the board. Sync. information is output on all three video signals.

1	Composite blank	Input/Output
2	Vertical Sync	Output
3	Composite or Horizontal Sync	Output
4	Blue	Output 75R
5	Green	Output 75R
6	Red	Output 75R

30.2 Graphics library software

The IMS B419 includes the IMS F003 graphics library software. This provides a two dimensional graphics library, functionally conforming with a subset of the Computer Graphics Interface (CGI) standard. This library is sufficiently flexible to allow it to be used as a building block to implement additional 2 or 3-dimensional graphics operations.

Functions in the library fall into a number of clearly defined areas. A family of functions exists for drawing points, lines, and arcs. Another family handles two dimensional drawing operations and fills. The text support provided by another family is expandable by the programmer to accommodate personalised font information. Two dimensional screen-pixel operations offer block copying, zooming, and rotation. An auxiliary category handles miscellaneous initialisation of the graphics card and data structures used by the rest of the library.

In order to separate those parts of the graphics software that are hardware dependent, a supplementary library for the IMS B419, called B419.LIB, provides framestore-specific initialisation, multi-frame display buffering, and colour control.

The CGI library will be useful in application areas including engineering drawing, mimic diagrams, interactive drawing, modelling and concept visualisation.

The IMS F003 CGI software has been implemented in ANSI C, using the INMOS TCOFF Toolset C compiler.

Of the two libraries provided, the main library CGILIB.LIB, handles all the CGI drawing operations which are independent of the graphics card type. It is supplied in binary form as a library, compatible with the INMOS TCOFF toolset. The supplementary library, B419.LIB, adapts the behaviour of the main CGI library to run on the IMS B419.

Summary of CGI Graphics Library functions

CGILIB.LIB functions			
cgi_addsptext	cgi_disjpolyline	cgi_polygon	cgi_setfont
cgi_addtext	cgi_dot	cgi_polyline	cgi_setlinestyle
cgi_arc	cgi_errstat	cgi_rect	cgi_setorient
cgi_arcc	cgi_fcircle	cgi_rot	cgi_setpelstyle
cgi_chrbegin	cgi_ffan	cgi_search	cgi_sptext
cgi_chrspace	cgi_fhline	cgi_setbcol	cgi_strokearc
cgi_chrz	cgi_frect	cgi_setdrawmode	cgi_text
cgi_circle	cgi_ftrap	cgi_setdrawscreen	cgi_zoom
cgi_cls	cgi_init	cgi_line	cgi_setfcol
cgi_copy	cgi_paint		cgi_setfillstyle
B419.LIB functions			
fs_displaybank	fs_initVTG	fs_setpalette	
fs_initscreen	fs_screenaddr		

Table 30.7 Summary of CGI Graphics Library functions

30.3 Mechanical details

Figure 30.2 indicates the vertical dimensions of a single IMS B419 and Figure 30.3 shows the outline drawing of the IMS B419.

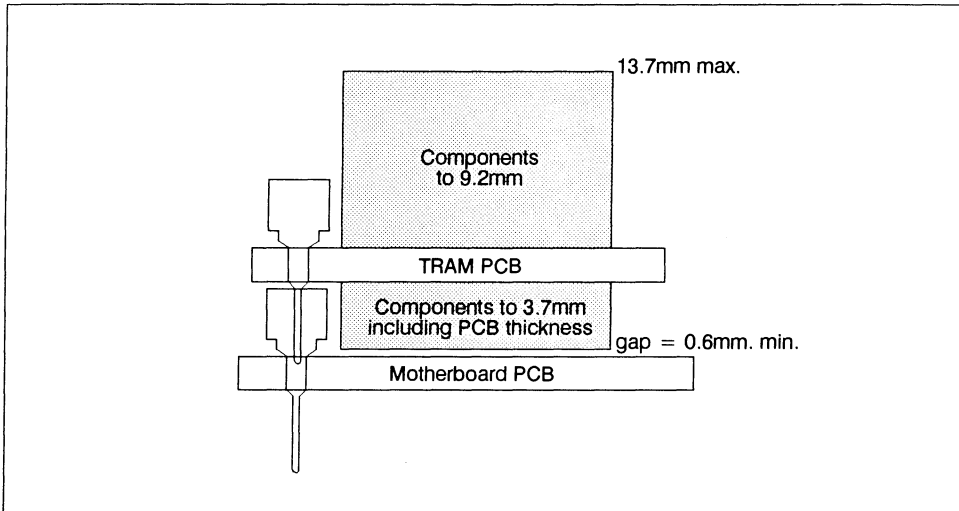


Figure 30.2 IMS B419 height specification

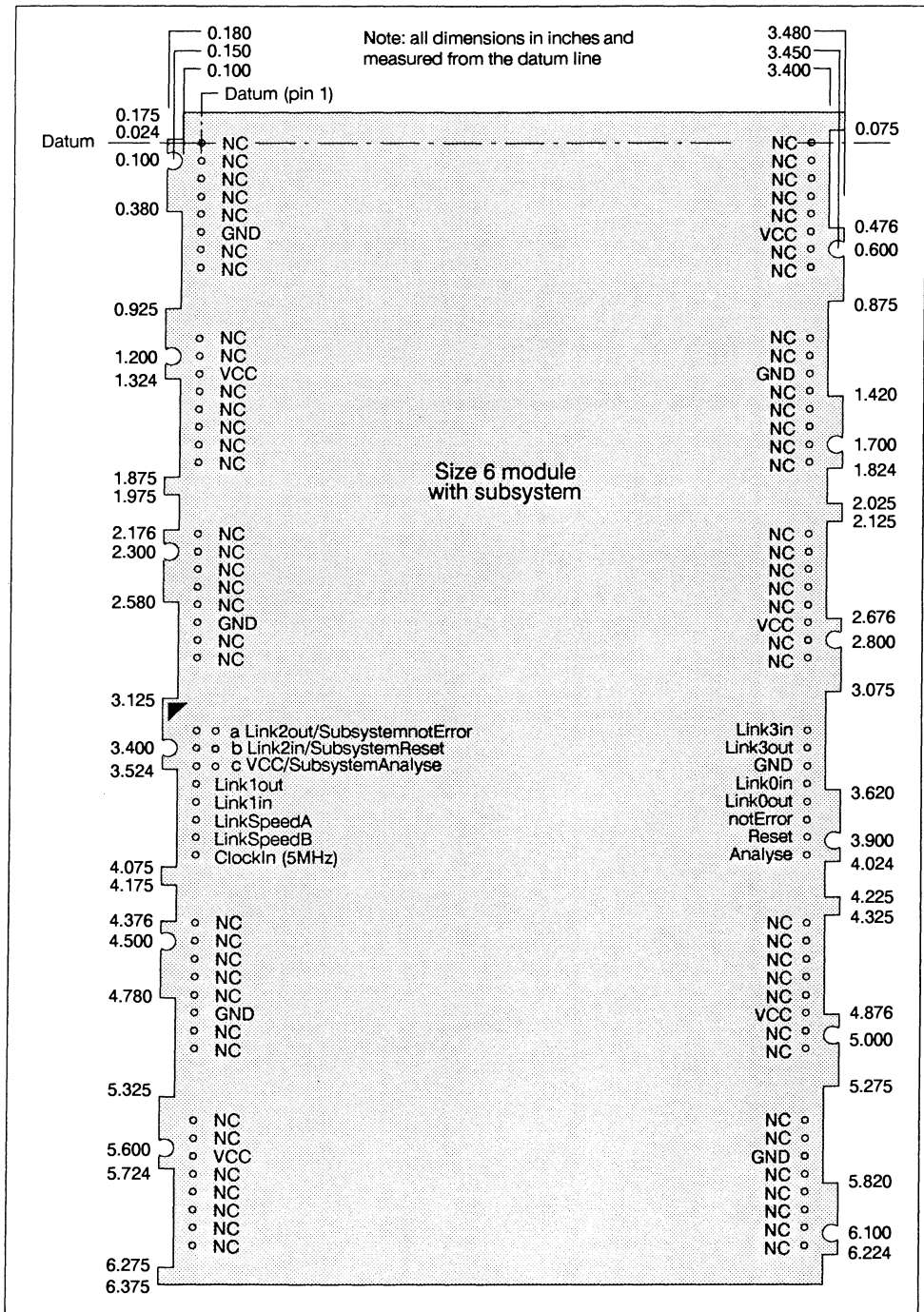


Figure 30.3

30.4 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
VCC, GND		Power supply and return	3,14
ClockIn	in	5 MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkSpeedA,B	in	Transputer link speed selection	6,7

Table 30.8 IMS B419 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
 - 2 Details of the physical pin locations can be found in Fig 30.3.
- LinkOut0-3** Transputer link output signals. These outputs are intended to drive into transmission lines with a characteristic impedance of 100Ω. They can be connected directly to the **LinkIn** pins of other transputers or TRAMs.
- LinkIn0-3** Transputer link input signals. These are the link inputs of the transputer. Each input has a 10kΩ resistor to **GND** to establish the idle state, and a diode to **VCC** as protection against ESD. They can be connected directly to the **LinkOut** pins of other transputers or TRAMs.
- LinkSpeedA, LinkSpeedB** These select the speeds of **Link0** and **Link1,2,3** respectively. Table 30.9 shows the possible combinations.

LinkSpeedA	LinkSpeedB	Link0	Link1,2,3
0	0	10 Mbits/s	10 Mbits/s
0	1	10 Mbits/s	20 Mbits/s
1	0	20 Mbits/s	10 Mbits/s
1	1	20 Mbits/s	20 Mbits/s

Table 30.9 Link speed selection

ClockIn A 5 MHz input clock for the transputer and CVC. The transputer synthesises its own high frequency clocks. **ClockIn** should have a stability over time and temperature of 200 ppm. **ClockIn** edges should be monotonic within the range 0.8V to 2.0V with a rise/fall time of less than 8 ns.

Reset Resets the transputer, and other circuitry. **Reset** should be asserted for a minimum of 100ms. After **Reset** is deasserted a further 100ms should elapse before communication is attempted on any link. After this time, the transputer on this TRAM is ready to accept a boot packet on any of its links.

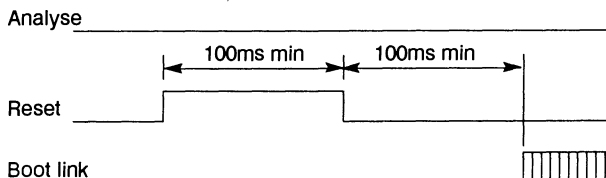


Figure 30.4 Reset timing

Analyse is used, in conjunction with **Reset**, to stop the transputer. It allows internal state to be examined so that the cause of an error may be determined. **Reset** and **Analyse** are used as shown in figure 30.5. A processor in analyse mode can be interrogated on any of its links.

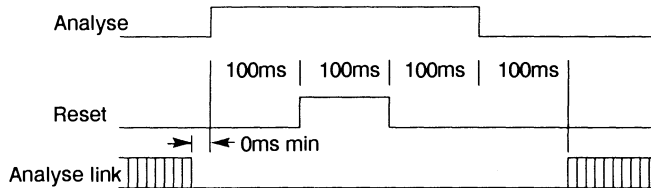


Figure 30.5 Analyse timing

notError An open collector output which is pulled low when the transputer asserts its Error pin. **notError** should be pulled high by a 10k Ω resistor to **VCC**. Up to 10 **notError** signals can be wired together. The combined error signal will be low when any of the contributing signals is low.

30.5 Specification

TRAM feature IMS B419-4		Unit	Notes
Transputer type	IMS T800-20		
Amount of DRAM	2	Mbyte	
Amount of VRAM	2	Mbyte	
DRAM/VRAM cycle time	200	ns	
Subsystem controller	Yes		
Peripheral circuitry	IMS G300B		
Size (TRAM size)	6		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	6.55	inch	1
Component height above PCB	13.7	mm	
Component height below PCB	3.7	mm	2
Weight	175	g	
Storage temperature	0-70	$^{\circ}$ C	
Operating temperature	0-50	$^{\circ}$ C	3
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	9	W	4

Table 30.10 IMS B419 specification

Notes:

- 1 An additional clearance of 1in. is required for the video connections.
- 2 This dimension includes the thickness of the PCB.
- 3 The figure quoted refers to the ambient air temperature.
- 4 The figure quoted has not been characterised and is subject to change.

30.6 References

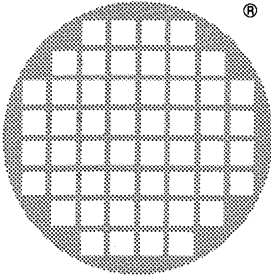
- 1 Crystal Oscillator Module (Appendix A.3), IMS B419-4 Graphics TRAM User Manual, Inmos Limited, 1990
- 2 Graphics Databook, 2nd Edition, Inmos Limited, 1990

30.7 Ordering Information

Description	Order Number
IMS B419 TRAM with IMS G300B	IMS B419-4*
IMS F003 2D graphics library	IMS F003A-1

*Includes IMS F003A-1 2D graphics library

Table 30.11 Ordering information



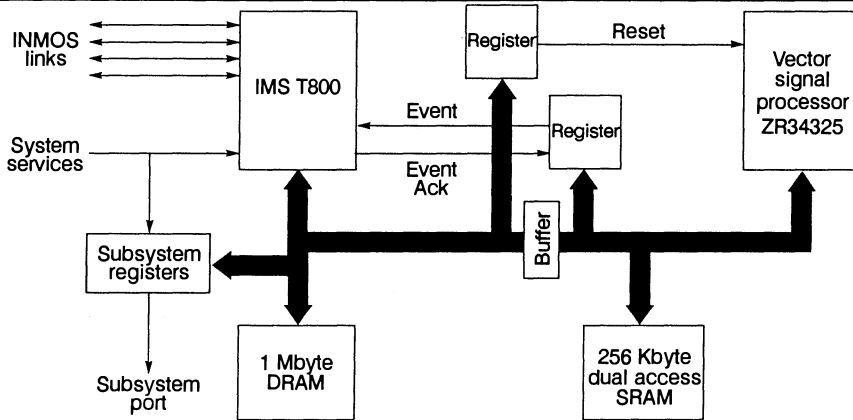
inmos[®]

IMS B420

Vector processing TRAM

Size 4

Engineering Data



FEATURES

- IMS T800 -25 or T800 -20 floating point transputer
- High performance vector/signal processing co-processor (ZR34325)
— e.g. 1K complex FFT < 2ms for 25MHz co-processor
- Both processors support IEEE 754-1985 floating point
- 4 INMOS serial communication links allowing connection of multiple VecTRAMs
- 1 Mbyte DRAM for IMS T800
- 256 Kbyte, dual access SRAM for full speed co-processor operation
- Size 4 TRAM
- Sub-system port
- Supplied with IMS F000 C and OCCam libraries
- IMS F007 DSP libraries available
- Designed to a published specification (INMOS Technical Note 29)

GENERAL DESCRIPTION

The IMS B420 VecTRAM is a transputer module combining the communications ability and scalar floating point performance of the IMS T800 with a high performance vector/signal processing co-processor (ZR34325). The two processors can operate concurrently, using separate dynamic and static memory blocks. The vector/signal processor is normally operated as a slave to the IMS T800 which can read and write to the SRAM, to set up vector/DSP routines as well as to load data for processing. The two processors can handshake via interrupts, thus allowing the transputer to initiate vector routines and the co-processor to signal the termination of the requested task.

Examples at 25MHz operation of the coprocessor's capabilities are: 1K complex FFT in 1.8 ms, 10×10 by 10×10 matrix multiplication in approximately 135 μ s and 64-tap FIR in 6 μ s.

Application areas include speech and image processing, graphics and numerical processing, radar, sonar and seismology.

31.1 Introduction

The IMS B420 (VecTRAM) has been developed to offer both scalar and vector processing in a single 'TRAM' module. This module combines the scalar floating/integer performance and general purpose capabilities of the transputer; with a high performance vector/signal processing co-processor. A set of software libraries is available which allows programs written in high level languages, running on the transputer, to call vector routines executed on the co-processor.

The IMS T800 transputer has sole access to 1 Mbyte of dynamic memory. 256 Kbyte of the static memory is accessible by both the transputer and the co-processor, under the control of an arbiter. The two processors can operate concurrently, using the separate dynamic and static memory blocks. The vector /signal processor is operated as a slave to the IMS T800 which can read and write to the static RAM, to set up vector/signal processing routines as well as to transfer data for processing.

The operation of the module is better understood with a description of the two processors.

The IMS T800 Processor

The IMS T800 transputer integrates a 12.5 MIPS (25MHz) 32-bit central processing unit(CPU), a 2 MFLOP scalar floating point unit (FPU) supporting IEEE arithmetic, communication links for parallel processing, and 4 Kbytes of on-chip memory. Each communication link is capable of providing around 1.2 Mbyte/s data rate in each direction. These allow easy connection to other TRAMs and transputers.

The IMS T800 also has one- and two-dimensional block-move capabilities. This allows blocks of data to be moved speedily from one memory segment to another. This is particularly important in the support for a vector co-processor where data blocks must be transferred for processing. The two-dimensional block-move, in particular, allows operations such as corner turning to be carried out while the data is being transferred.

The ZR34325 Vector/Signal Processor

The ZR34325 is a vector/signal processor which supports vector floating point arithmetic with a peak performance of 37.5 MFLOP. The floating-point arithmetic is 32-bit and conforms to the IEEE 754-1985 standard, making it compatible with the single-precision floating point format on the transputer. The device is optimised to execute efficiently a wide variety of signal/vector processing functions. These include multidimensional FFT's, digital filters, vector-scalar, vector-vector, and matrix operations. The block diagram of the processor, showing various functional blocks, is given in figure 31.1.

The co-processor integrates six main functional units plus on-chip memory and registers. The main functional units are:

- Execution Unit:** This unit is configured to efficiently execute complex multiply/accumulate operations, while conforming to the IEEE standard for binary floating-point arithmetic. All four rounding modes, as well as $\pm\infty$, NaNs and denormalised numbers are supported. This unit can operate on both internal and external memories. The results can be written in the on-chip RAM or registers. The intermediate or final results which are written to on-chip registers can be automatically copied to off-chip memory. The execution unit consists of three major computational blocks; a 32-bit floating-point multiplier, a 32-bit floating-point adder, and a second adder/subtractor which can perform accumulation in extended 44-bit precision. The execution unit also contains address-generation hardware for accessing internal RAM, and coefficient ROM. It also includes three registers. Among operations supported by the execution unit are vector multiply, vector addition, vector subtraction, comparisons, as well as direct support for higher level functions such as FFT's, matrix operations, and filtering.
- Bus-Interface Unit:** This unit coordinates the activities of the external memory interface. It supports a 32-bit bidirectional data bus, and a 24-bit address bus allowing a 16 Mword address space. The interface also allows master or slave operations. In master mode the co-processor

has control of the external memory and it fetches its own instruction. In slave mode, an external host can read to/write from the memory-mapped internal registers and memory.

- **Move Unit:** This unit transfers data between internal and external memories. It supports a variety of vector-oriented addressing including: 1-D/2-D transfers, complex and real data addressing, circular buffers, bit-reversal. It also transfers data between the external memory and the execution unit via the vector unit.
- **Vector Unit:** This unit acts as a vector buffer (FIFO) between the move unit and the execution unit. It decouples the execution unit from the external memory behaviour thus improving performance.
- **Control Unit:** This unit supports external memory address computations, and also performs 24-bit integer arithmetic and logical operations on the integer registers. It also carries out all load and store operations on all integer registers.
- **Fetch Unit:** This unit manages an instruction queue that is implemented as a FIFO with four 64-bit words. Every instruction fetched goes through this buffer except program flow control instructions such as LOOP. This allows loop code to be kept in the instruction buffer during the first pass and used subsequently.

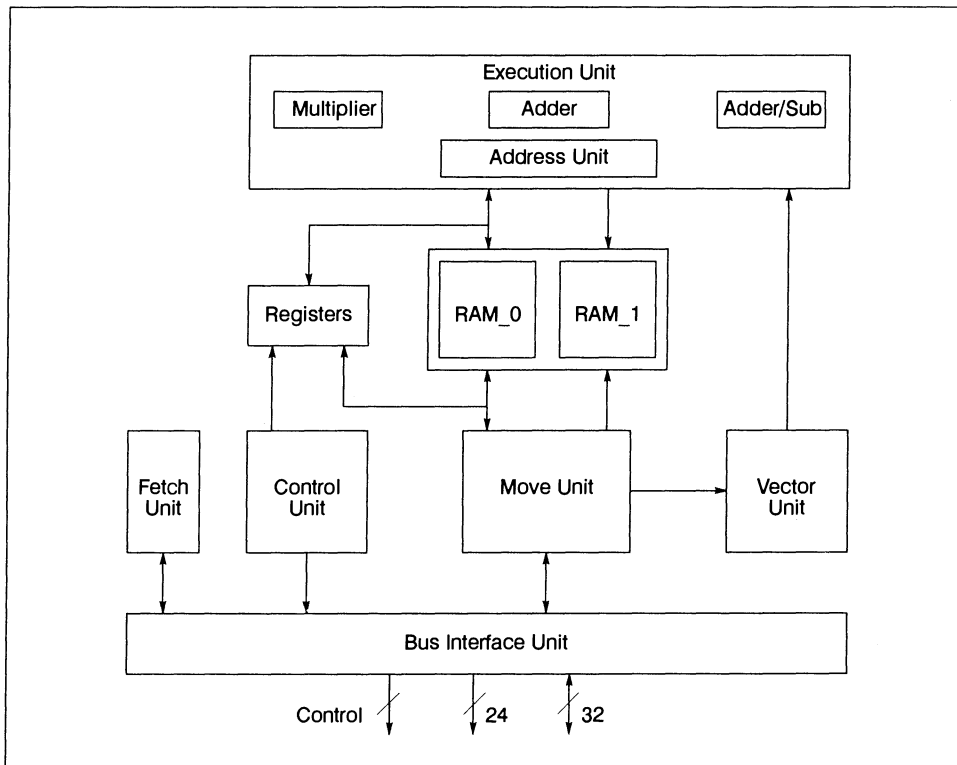


Figure 31.1 The signal/vector processor block diagram

Other major elements within the co-processor are the internal RAM and registers. The internal RAM consists of 512 bytes of memory organised as an array of 64 complex words. The RAM can also be configured as two separate blocks, each of 32 complex words. Operands in the internal RAM can be either complex or real vectors. To increase internal RAM utilisation, double length real vectors can use both real and imaginary parts of the RAMs.

31.2 Transputer memory map

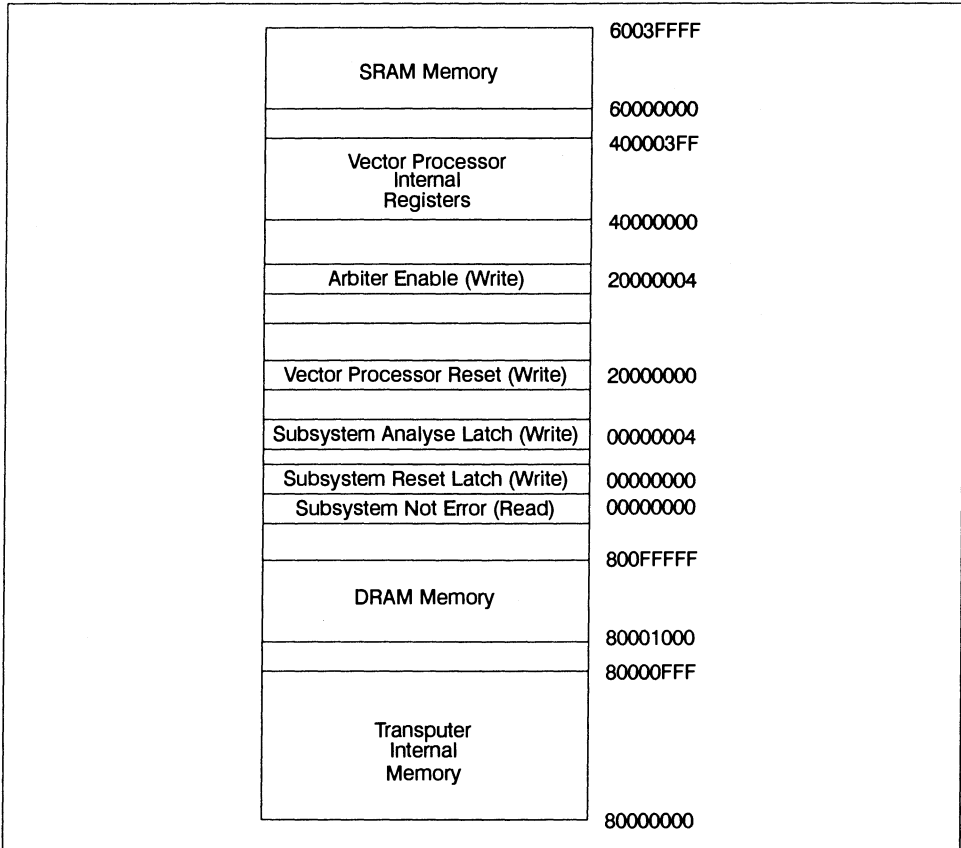


Figure 31.2 Transputer memory map (byte addresses)

- **Vector Processor Reset:** To reset the vector processor, write 0 to this location and wait for at least 1 μ s. Then write 1 to deassert the reset signal.
- **Arbitr Enable:** Write 1 to enable the SRAM access arbiter. In this mode, both the IMS T800 and the ZR34325 can access the SRAM. Cycle time from the IMS T800 is typically 6 clock cycles (300ns at 20MHz). Write 0 to disable the SRAM access arbiter. In this mode, only the IMS T800 may access the SRAM. This mode provides faster access for block copying of data and programs, while the ZR34325 is not running. Cycle time from the IMS T800 is 4 clock cycles (200ns at 20MHz). The IMS T800 always accesses the DRAM in 4 clock cycles (200ns at 20MHz).

31.3 ZR34325 memory map

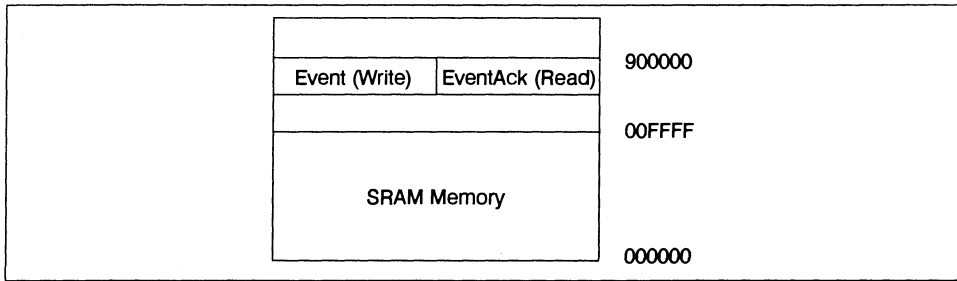


Figure 31.3 ZR34325 memory map (word addresses)

- **Event:** Write 1 to assert the **Event** pin of the IMS T800; write 0 to deassert it. Reading this location returns the state of the the **EventAck** pin of the IMS T800.

31.4 Mechanical details

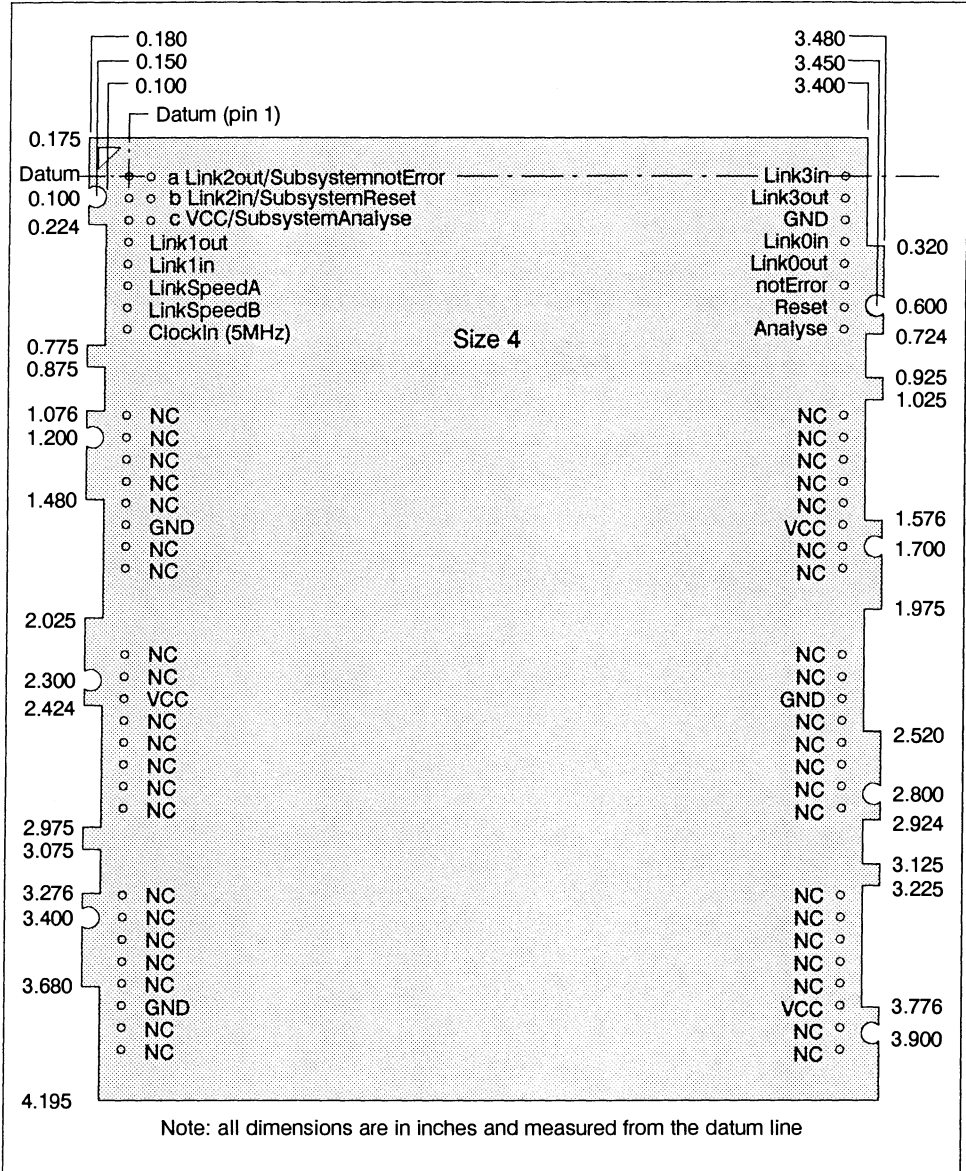


Figure 31.4 IMS B420 outline drawing (All dimensions in inches)

31.5 Specification

TRAM feature		Unit	Notes
IMS T800 transputer	1		
ZR34325 vector processor	1		
Fast dual-port RAM	256	Kbyte	
DRAM	1	Mbyte	
TRAM size	4		
Length	3.66	inch	
Width	4.35	inch	
Pitch between pins	3.30	inch	
Component height above PCB	9.2	mm	
Component height below PCB	3.7	mm	1
Weight (approx.)	113	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	
Power supply voltage (Vcc)	4.75-5.25	Volt	
Power consumption (Max)	9.5	W	2

Table 31.1 IMS B420 specification

NOTES

- 1 This dimension includes the thickness of the PCB.
- 2 Measured at Vcc = 5.25V.

31.6 IMS F000 software library

31.6.1 Software support

Currently INMOS provides a C signal/vector processing library (IMS F000A) which allows the co-processor to be used from a high level language running on the transputer. This library is normally supplied in binary and compatible with the parallel C compiler family, IMS x11. Future releases of this software will extend to other compiler families, in particular, the new generation of INMOS C and OCCam compilers and will support additional functions.

The IMS F000 consists of a library of C functions which implement common vector/signal processing tasks. The functions are callable from a C program running on an IMS B420 TRAM, and can be used to dramatically speed up parallel applications and system performance involving vector/signal processing computation. The use of the libraries is most easily shown by a simple example:

```
#include"decc11.h"      /*Function declarations */
#include"memc11.h"      /*Workspace constants */

main()
{
float a[100], b[100], c[100];
int i, flag;

/* Initialise the co-processor and set up workspaces*/
VT_INIT(co-processor_MEM_BASE, LIB_WORKSPACE_BASE,
USER_WORKSPACE_BASE, co-processor_MEM_TOP);

/* Initialise test data arrays */
for(i=0; i<100; i++){
    a[i] = 10.0;
    b[i] = 20.0;
}

flag =0;

/* Call vector multiply function */
VT_MULT_F32R(a, 1, b, 1, c, 1, 100, flag);
}
```

During program execution, when a vector library function is called, it first checks the addresses of its operands. If the data to be processed is in the transputer local memory space, it is automatically copied (using the blockmove capability of the T800) to a predetermined area in the co-processor space. The co-processor is then activated to execute the required function.

If the destination vector operand address, specified in the call, is in the transputer space, the processed data is automatically copied back to the specified area in the transputer memory space. This built-in copying means that the co-processor operation can be totally transparent to the programmer, and programs can be accelerated without the need for detailed knowledge about the operation of the IMS B420 TRAM.

The overhead associated with data copying between the transputer and co-processor (or visa versa) is avoided if the source and destination operands for the specified function are already in the co-processor local memory space. The library functions automatically check the operand addresses and take appropriate action. In general, if the address of an input or an output operand, in a function call, is in the co-processor space, no data copying will take place for that operand. This is particularly important if operands are to undergo several vector/signal processing operations. For optimal performance the user can specify desti-

nation (and/or source) addresses which are local to the co-processor address space. In this way data copying, between the transputer and the shared memory area can be minimised.

Apart from vector and arithmetic functions, the IMS F000 includes efficient vector move functions which allow optimisation at the application level. The library also supports co-processor control calls which are used to set rounding modes and error handling.

IMS F000A function calls include:

VT_ABS_F32R

Function: Vector Absolute Value – Real

VT_ADD_F32R

Function: Vector Addition – Real

VT_CMP_F32R

Function: Vector Compare – Real

VT_DISERROR

Function: Disable co-processor Error Flags

VT_DOT_F32C

Function: Vector Dot Product – Complex

VT_DOT_F32R

Function: Vector Dot Product

VT_ENERROR

Function: Enable co-processor Error Interrupts

VT_FFT_F32C

Function: Fast Fourier Transform – Complex

VT_F32TOI16_R

Function: Vector Floating point to Integer (16-bit) Conversion

VT_IFFT_F32C

Function: Inverse Fast Fourier Transform – Complex

VT_I16TOF32_R

Function: Vector Integer (16-bit) to 32-bit floating point Conversion

VT_LOG10_F32R

Function: Vector Log to the base 10 – Real

VT_MAG_F32C

Function: Vector Magnitude – Complex

VT_MAGSQ_F32C

Function: Vector Magnitude Square – Complex

VT_MAX_F32R

Function: Find the Element with the Maximum Value and its position – Real

VT_MEAN_F32R

Function: Vector Mean – Real

VT_MIN_F32R

Function: Find the Element with the Minimum Value and its position – Real

VT_MOV_BYTE

Function: Vector Move – Bytes

VT_MOV_WORD

Function: Vector Move – Words (32-bit)

VT_MULT_F32C

Function: Vector Multiply – Complex

VT_MULT_F32R

Function: Vector Multiply – Real

VT_POWER_F32R

Function: Vector Power – Real

VT_ROUNDMODE

Function: Modify co-processor Rounding mode

VT_SCALE_F32R

Function: Vector Scale – Real

VT_SQRT_F32R

Function: Vector Square Root – Real

VT_SUB_F32R

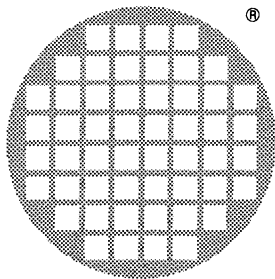
Function: Vector Subtract – Real

31.7 Ordering Information

Description	Order number
IMS B420 VecTRAM 20 MHz operation	IMS B420-3*
IMS B420 VecTRAM 25 MHz operation	IMS B420-5*
IMS F000 VecTRAM library software – supplied on IBM PC format 5 1/4" and 3 1/2" discs.	IMS F000A-1
IMS F007 DSP software library – supplied on IBM PC format 5 1/4" and 3 1/2" discs.	IMS F007A-1

Table 31.2 Ordering information

* Includes IMS F000A-1 VecTRAM library software



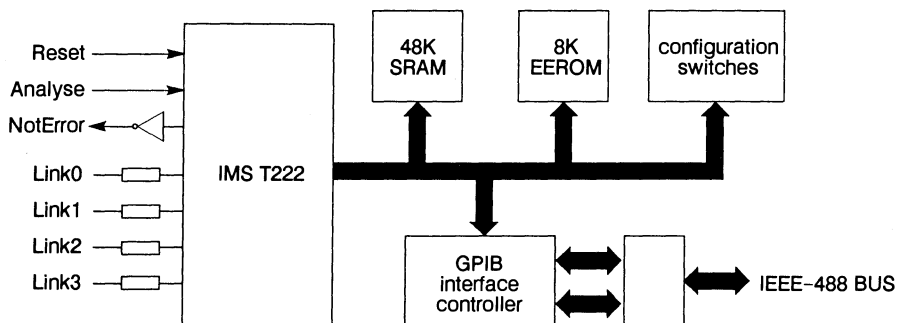
inmos[®]

IMS B421

IEEE 488 GPIB

Size 4 TRAM

Engineering Data



FEATURES

- IMS T222 transputer
- 48 Kbytes of two-cycle RAM
- Full electrical compliance with IEEE-488 specification
- Size 4 TRAM
- Switchable GPIB bus address
- On-board non-volatile storage for configuration data
- Communicates via 4 INMOS links
- Supplied with IMS F001 GPIB library software
- Designed to a published specification (*INMOS Technical Note 29*).

GENERAL DESCRIPTION

The GPIB TRAM allows IEEE-488 test and instrumentation systems to be directly connected to networks of transputers. The parallel interface permits high speed communication of control and measurement information, and the power of the transputer can provide sophisticated data analysis facilities. The user can define the characteristics of the GPIB interface in terms of address, etc., for maximum flexibility in system configuration.

32.1 Description

The IMS B421 TRAM combines an IEEE-488 interface with an INMOS transputer. The hardware design follows the electrical requirements of IEEE-488.1, while the companion software package IMS F001 provides support for the data transfer requirements of IEEE-488.1. Used together, these two products provide a bridge between the sophisticated data acquisition capabilities of GPIB instrumentation, and the processing power of transputer networks. The IMS B421 has been designed for maximum flexibility, to allow its use in the widest possible range of applications. As an example of a small system, the TRAM could be embedded in an instrument, with the onboard transputer carrying out all processing and control tasks. In a larger system, the TRAM might be used to convey arrays of data to a network of transputers for complex high-speed processing.

32.1.1 The IEEE-488 standard

The IEEE-488 standard defines a means of interconnecting electronic instruments to form a system around which data and control information may be passed. Such an instrument system can be programmed to perform complex sequences of data acquisition and equipment control. Since the advent of cheap microprocessors, many instruments equipped with IEEE-488 interfaces have become available, covering a very broad range of functions and performance levels. In its original form, the standard described only the means by which messages could be exchanged via the interface; the format of these messages, for example the actual character string used to represent the numeric value of a reading, was at the discretion of the instrument designer. So, two instruments might be IEEE-488 compatible in the sense that they could exchange individual characters as defined in the standard, and yet be unable to decode each other's numeric formats. The original standard has therefore been expanded; IEEE-488 now refers collectively to IEEE-488.1, the original low-level specification, and to IEEE-488.2, which defines much more closely the formats and sequences of messages passed between the instruments in a system.

The standard has been adopted and supported by many manufacturers, whose implementations have been identified by proprietary names too numerous to list. Probably the most common name for the interconnection system defined by the standard is General Purpose Interface Bus, often abbreviated to GPIB as in this document.

It is assumed that users of the IMS B421 already have some familiarity with the IEEE-488 standard, and in any case it is beyond the scope of this document to provide a description of its workings. IMS B421 users who do not already have access to copies of the IEEE-488 standard documents are strongly recommended to obtain them; useful information on this appears below. IMS B421 users seeking an introduction to the standard are encouraged to obtain a copy of the Tutorial Introduction to the HPIB.

- 1 IEEE standard 488.1-1987, 'Digital Interface for programmable instrumentation'. Available from The IEEE Inc., 345 East 47th St., New York, NY 10018.
- 2 IEEE standard 488.2-1987, 'Codes, Formats, Protocols, and Common Commands for use with ANSI/IEEE Std 488.1-1987'. Available from The IEEE Inc., 345 East 47th St., New York, NY 10018.
- 3 'Tutorial Description of the Hewlett-Packard Interface Bus'.

For availability consult your nearest Hewlett-Packard sales office.

32.1.2 Companion software package IMS F001

This software provides a convenient means of access to the facilities of the TRAM via library calls, so that the user need not be concerned with low-level hardware details. The product is intended to perform all commonly required GPIB handling, to minimise the software effort required of the user.

Note that the software is not resident on the board in any permanent sense; no executable code is held in non-volatile storage on the IMS B421 as supplied. When the TRAM is powered up, the onboard transputer enters the boot-from-link condition, meaning that it expects bootstrap code to be supplied via one of its

four serial links. Until it has loaded suitable code, the IMS B421 cannot respond to, or originate, any GPIB activity, nor can it perform any processing tasks.

The IMS F001 software communicates via an input channel and an output channel. This allows the user two different approaches to IMS B421 software implementation.

Firstly, the channels may be directly mapped to one of the four pairs of hardware serial links in the onboard transputer, with the combination of the IMS F001 and IMS B421 treated as a 'black box' add-on to a transputer network. In this case, IMS F001 is the only software running on the IMS B421's onboard transputer.

Secondly, the channels may be implemented as soft links between IMS F001 and a user program running concurrently on the onboard transputer. This approach allows the IMS B421 to perform some front-end processing such as buffering, or even to provide the entire processing resource. A later section provides hardware information which will be of use to programmers who intend to produce such concurrent programs.

32.2 Pin descriptions

Pin	In/Out	Function	Pin No.
System Services			
Vcc, GND		Power supply and return	3,14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13,5,2,16
LinkOut0-3	out	INMOS serial link outputs from transputer	12,4,1,15
LinkSpeedA,B	in	Transputer link speed selection	6,7
Subsystem Services			
SubsystemReset	out	Subsystem reset	1b
SubsystemAnalyse	out	Subsystem error analysis	1c
SubsystemnotError	in	Subsystem error indicator	1a

Table 32.1 IMS B421 Pin designations

Notes:

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in figure 32.6.

32.2.1 Standard TRAM signals

LinkOut0-3 Transputer link output signals. These outputs are intended to drive into transmission lines with a characteristic impedance of 100Ω. They can be connected directly to the **LinkIn** pins of other transputers or TRAMs.

LinkIn0-3 Transputer link input signals. These are the link inputs of the transputer on the IMS B407. Each input has a 10kΩ resistor to **GND** to establish the idle state, and a diode to **VCC** as protection against ESD. They can be connected directly to the **LinkOut** pins of other transputers or TRAMs.

LinkSpeedA, LinkSpeedB These select the speeds of **Link0** and **Link1,2,3** respectively. Table 32.2 shows the possible combinations.

LinkSpeedA	LinkSpeedB	Link0	Link1,2,3
0	0	10 Mbits/s	10 Mbits/s
0	1	10 Mbits/s	20 Mbits/s
1	0	20 Mbits/s	10 Mbits/s
1	1	20 Mbits/s	20 Mbits/s

Table 32.2 Link speed selection

ClockIn A 5MHz input clock for the transputer. The transputer synthesises its own high frequency clocks. **ClockIn** should have a stability over time and temperature of 200ppm. **ClockIn** edges should be monotonic within the range 0.8V to 2.0V with a rise/fall time of less than 8ns.

Reset Resets the transputer, and other circuitry. **Reset** should be asserted for a minimum of 100ms. After **Reset** is deasserted a further 100ms should elapse before communication is attempted on any link. After this time, the transputer on this TRAM is ready to accept a boot packet on any of its links.

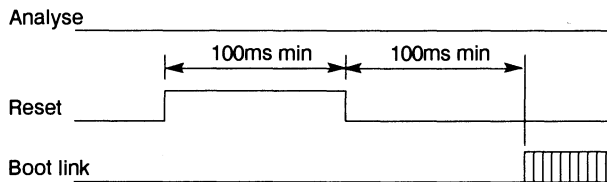


Figure 32.1 Reset timing

Analyse is used, in conjunction with **Reset**, to stop the transputer. It allows internal state to be examined so that the cause of an error may be determined. **Reset** and **Analyse** are used as shown in figure 32.2. A processor in analyse mode can be interrogated on any of its links.

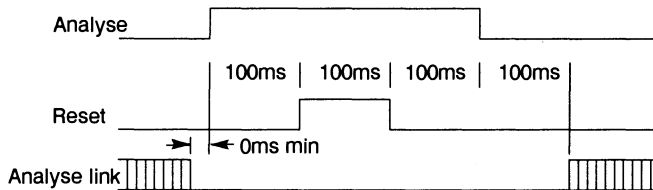


Figure 32.2 Analyse timing

notError An open collector output which is pulled low when the transputer asserts its Error pin. **notError** should be pulled high by a 10k Ω resistor to **VCC**. Up to 10 **notError** signals can be wired together. The combined error signal will be low when any of the contributing signals is low.

32.2.2 Subsystem signals

The IMS B421 has a subsystem port in addition to the usual TRAM signals. This enables the TRAM to reset or analyse a subsystem of other TRAMs and/or motherboards. The polarity of these signals is the same as that of the **Reset**, **Analyse** and **notError** standard TRAM signals. Therefore, the IMS B421 subsystem can drive other TRAMs on the same motherboard with no intermediate logic. However, **SubsystemReset** and **SubsystemAnalyse** must go through inverting buffers if they are to drive a subsystem off the motherboard.

These subsystem signals are accessed by writing or reading to control registers in the transputer memory space. See section 32.6.1.

32.3 Hardware features

This section gives a general description of the facilities provided on the IMS B421. All users are recommended to read it, to gain familiarity with the product.

32.3.1 Onboard transputer system

The IMS B421 TRAM has an IMS T222-20 transputer with 4K bytes of fast internal RAM. This is supplemented by 48K bytes of external static RAM, which runs without wait states. The TRAM is thus provided with considerable processing power in comparison with many existing IEEE-488 interface products, allowing it to provide a compact solution in embedded applications which might otherwise require separate interface and processing modules.

32.3.2 IEEE-488 Interface

This is provided by a Texas Instruments TMS9914A GPIB controller, in conjunction with SN75160B and SN75162A buffers. These devices allow the IMS B421 to act as a System Controller, non-system Controller, Talker or Listener, and ensure full electrical compliance with the IEEE-488 standard. Note that the initial release of IMS F001 only supports operation of the IMS B421 as System controller or as a Talker/Listener.

32.3.3 Electrically Erasable Read Only Memory (EEROM)

The IMS B421 TRAM contains an EEROM device of 8K byte capacity. This is provided essentially to assist in implementing the requirements of IEEE-488.2, which calls for compliant devices to accept, retain and return various identifying information upon demand. The content of these messages cannot be determined in advance by INMOS, so the EEROM is provided as a non-volatile means of retaining the necessary character strings, which may be conveniently stored in the device by IMS F001 commands. The device capacity is more than enough to store the information required for compliance with the standard, so the remainder may be allocated to any purpose defined by the user, and is easily accessed via additional IMS F001 commands.

32.3.4 Power-up/Power-fail detection

IEEE-488.2 defines a standard format in which a compliant device must be able to report its status, including an indication as to whether it is reporting status for the first time since being powered up. The IMS B421 hardware supports this by means of a supply voltage monitor, which (via the IMS F001 software) causes the appropriate bit of the status information to be set at power up. The bit is cleared immediately after being read, so that subsequent status reports show no power-up indication as long as the power supply remains within specified limits.

The hardware is also configured to reset the IMS B421's onboard transputer and any attached subsystem whenever the supply voltage rises above the specified minimum, to ensure predictable system behaviour following power-up or power failure.

32.3.5 Jumpers

Various characteristics of IMS B421/F001 operation are user-selectable. The selection is indicated via the pairs of pins on the top of the TRAM, by the presence or absence of the jumper connectors.

The jumper pins are not wired directly to the hardware functions to which they relate, but are read by software and relayed to hardware as required. This arrangement gives the software ultimate control over the relevant operating characteristics, so a configuration read from the jumpers may be modified in response to subsequent commands. For example, the GPIB address to which the IMS B421 responds may be altered by software command; it is not necessary to gain access to the jumper area of the TRAM and alter the jumpers' physical settings.

Section 32.5 gives details of the option selection jumpers.

32.4 Connector pin assignments

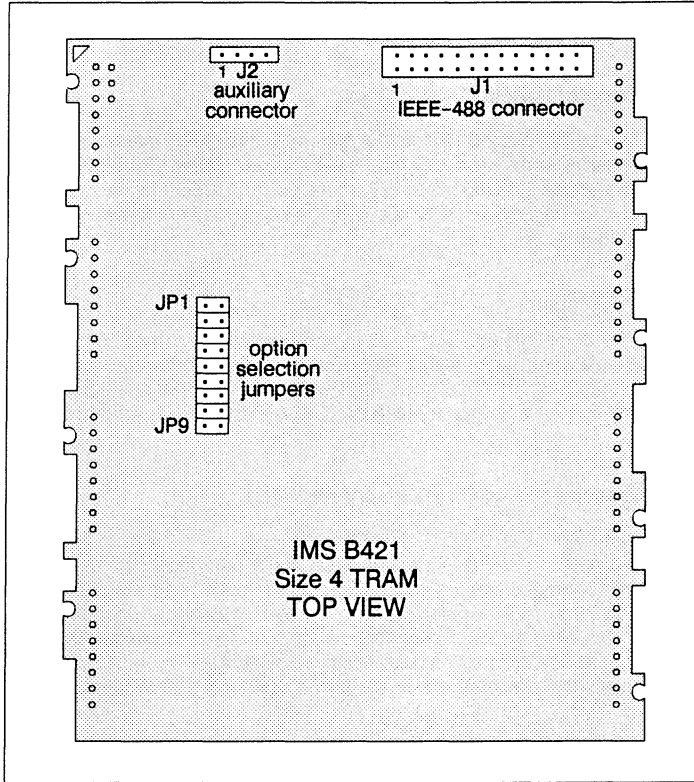


Figure 32.3 Location of pins and jumpers

32.4.1 IEEE-488 connector J1

The IMS B421 IEEE-488 interface is brought out via a standard 26 way pin header, J1. This has two rows of 13 pins on 0.1 inch pitch.

The mating IDC sockets and appropriate flat ribbon cable are widely available at low cost. Connectors which are mechanically compatible with IEEE-488 equipment are also available in IDC versions for use with the same type of ribbon cable.

Therefore, the simplest connection between a IMS B421 and a GPIB system is simply a length of ribbon cable with a 26-way dual-row 0.1 in. IDC socket at one end, and an IEEE-488 compatible IDC connector at the other. It is stressed that such a connection minimises performance as well as cost, so cable length must be kept as short as possible. It is only suitable for uses such as bringing out the IMS B421 interface to a panel-mounted connector, or making a temporary connection to a system during development.

Viewed from the component side of the IMS B421, i.e. looking at the contact pins of J1, the contact numbering is as below. Pin number 1 is identified by a small yellow dot, in the relative position indicated by the asterisk.

2	4	6	8	...	24	26
1	3	5	7	...	23	25

*

IMS B421 J1 Pin	IEEE-488 signal name	IEEE-488 compatible connector pin number
1	DIO1	1
2	DIO5	13
3	DIO2	2
4	DIO6	14
5	DIO3	3
6	DIO7	15
7	DIO4	4
8	DIO8	16
9	EOI	5
10	REN	17
11	DAV	6
12	Gnd	18
13	NRFD	7
14	Gnd	19
15	NDAC	8
16	Gnd	20
17	IFC	9
18	Gnd	21
19	SRQ	10
20	Gnd	22
21	ATN	11
22	Gnd	23
23	Gnd	12
24	Gnd	24
25	Gnd	(Note 1)
26	Gnd	(Note 1)

Table 32.3 J1 signal assignment

Notes:

- 1 The IEEE-488 standard defines only 24 signal and return conductors; pins 25 and 26 on the IMS B421 26-way connector are not used.
- 2 In the IEEE-488.1 standard, clause 23.3.1 calls for the use of shielded cable. This shield should be connected to a suitable point on the equipment enclosure in which the IMS B421 is installed, rather than to the IMS B421 itself.
- 3 **IMPORTANT:** The correspondence between contact numbers for the 26-way header and the IEEE-488 compatible connector is NOT one-to-one. In other words, a correctly assembled cable will generally NOT connect pin number N on the 26-way header to pin number N on the IEEE-488 compatible connector. This is because the IEEE-488 compatible connector contacts are numbered in a different pattern, which does not preserve the direct relationship of cable conductor number to contact number. The correct correspondence is shown in the column at the right of table 32.3.

32.4.2 Auxiliary connector J2

The IMS B421 also carries a smaller connector, J2, with four pins arranged in a single row on 0.1 inch pitch. Pin 1 of J2 is marked with a small yellow dot; pin numbering proceeds along the row of contacts.

Connection to J2 may be made with an IDC connector and ribbon cable, as described for J1 above; the termination at the far end of the cable is at the user's discretion.

IMS B421 J2 Pin	Signal name
1	IEEE-488 SRQ line status
2	IEEE-488 IFC line status
3	IEEE-488 REN line status
4	TRIG

Table 32.4 J2 signal assignment

The status of the three IEEE-488 lines is provided so that hardware in which the IMS B421 is embedded may directly observe Service Request messages, Interface Clear messages, and Remote/Local status. Note that the J2 pins have the same logic polarity as the IEEE-488 lines, i.e. TTL logic 0 level indicates TRUE. The TRIG signal is produced by the IMS B421 when it receives a Group Execute Trigger message. The pin goes to TTL logic 1 level to indicate this event, and may be used to trigger other embedded functions such as some form of data acquisition hardware.

32.5 Option selection jumpers

This section should be read before attempting to install the IMS B421 in any system, to avoid the possibility of incorrect setup.

The jumper links allow the user to configure various aspects of IMS B421/F001 operation, as detailed in the following subsections. See figure 32.3; jumpers for the location of the jumpers on the TRAM.

IMPORTANT: the configuration set up on the jumpers has no effect on TRAM operation until and unless the user issues commands which cause the IMS F001 software to read and adopt it. This applies to all nine jumpers.

32.5.1 Bus address jumpers, JP1 to JP5

The intended address value is indicated as a binary number on these jumpers. The encoding considers bit significance to begin at JP1 and increase in numeric sequence to JP5; presence of a jumper forces a bit's contribution to zero, whereas absence allows it to contribute its weighted value. For example, address 21₁₀ is encoded as follows:

Jumper	Status	Bit Significance	Contribution
JP1	Absent	1	1
JP2	Present	2	(none)
JP3	Absent	4	4
JP4	Present	8	(none)
JP5	Absent	16	16
	Address		16 + 4 + 1 = 21

IMPORTANT: Although the value 31₁₀ could be encoded by leaving all five jumpers JP1 to 5 absent, this is not a valid address in the IEEE-488 standard. Therefore, the IMS F001 software will give an error indication if the user attempts to read the jumpers with this address value set up.

32.5.2 Device capability jumpers, JP6 and 7

The IEEE-488 capability of the device may be selected via these jumpers according to the following table. (Refer to the standard for an explanation of the selections available).

JP7	JP6	Capability
Present	Present	Device, talk & listen
Present	Absent	Device, talk only
Absent	Present	Device, listen only
Absent	Absent	Controller, talk & listen

IMPORTANT: Operation of the IMS B421 as System Controller is also possible. This selection can only be made via software; refer to the IMS F001 documentation for further details.

32.5.3 Bus drive selection jumper, JP8

When this jumper is present the IMS F001 software will configure the IMS B421 for open-collector drive of IEEE-488 data signals. When the jumper is absent, tri-state drive is selected.

IMPORTANT: Note that tri-state drive **MUST** be used for high-performance applications; refer to clause 31 in the IEEE-488.1 standard for more details on data rate considerations. By contrast, open-collector drive **MUST** be used if the system is to make use of parallel-polling commands; it should be noted that the initial release of IMS F001 does not include support for parallel-polling.

32.5.4 Data protect jumper, JP9

The IMS F001 software takes the presence of this jumper to indicate that EEROM contents are **NOT** protected; IMS F001 commands which involve modification of EEROM contents will be accepted and obeyed.

Conversely, absence of this jumper is taken to mean that EEROM contents should not be altered. Any IMS F001 command which seeks to perform such an alteration will be rejected; refer to the IMS F001 documentation for more detail.

IMPORTANT: The absence of this jumper does **NOT** prevent access to the EEROM at hardware level. Protection is only implemented within the IMS F001 software with respect to its own commands, and will not intercept attempts by user code to perform direct writes to addresses within EEROM. The user is in any case strongly recommended to perform all EEROM access via IMS F001 commands, since EEROM writes must be performed subject to various operational constraints imposed by the device technology.

32.6 Hardware Information for programmers

This section provides essential information for programmers who intend to produce their own code to run on the IMS B421 hardware concurrently with the IMS F001 software. It is strongly recommended that any access to IMS B421 hardware functions required by user code be performed via the IMS F001 software. Direct access to the hardware facilities of the TRAM should only be attempted by experienced programmers; detailed descriptions of the operation of onboard devices are beyond the scope of this document, and should be sought in the relevant device manufacturer's data.

32.6.1 Memory configuration

Figure 32.4 shows a memory map for the system.

Internal RAM may be accessed in a single processor cycle, whereas external RAM requires two cycles. Where optimum performance is needed, the programmer should seek to compile code to internal RAM.

Subsystem register locations

The subsystem registers are not implemented in the standard way on the IMS B421, but are part of the I/O ports. Details are given in the sections 32.6.2 and 32.6.3

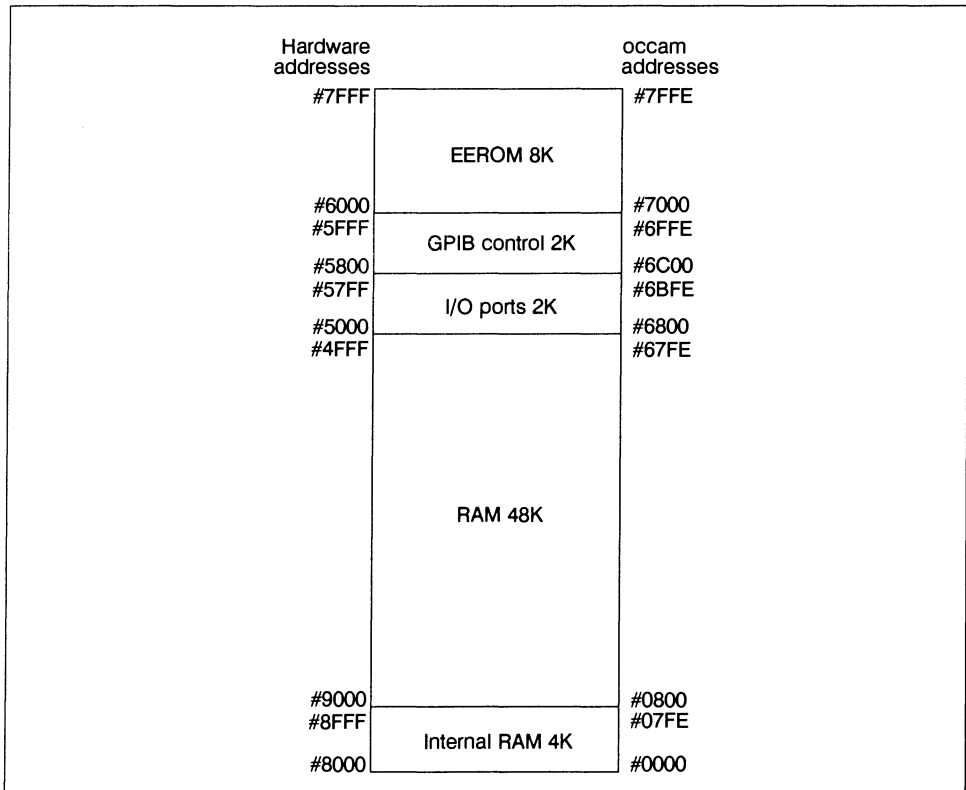


Figure 32.4 Memory map

32.6.2 Input port assignments

The I/O ports allow the reading of status on 16 input lines as one word, appearing at hardware address #5000. The address decoding is such that the same word may also be read at hardware addresses #5002, #5004, etc. up to and including #57FE. The usage of the input bits, beginning with the least significant, is as follows:

Input Bit	Purpose
0	Read back System Controller signal, sent from output port to TMS9914A controller. Bit read as one indicates that TMS9914A is to act as System Controller.
1	Read back Drive Type signal, sent from output port to GPIB buffers. Bit read as zero indicates that open collector drive is selected; bit read as one indicates tri-state drive.
2	Read back SubsystemAnalyse signal, sent from output port. Bit reads non-inverted logic level of control line.
3	Read back SubsystemReset signal, sent from output port. Bit reads non-inverted logic level of control line.
4	Controller in charge signal from TMS9914A. Bit read as zero indicates that the TMS9914A currently controls GPIB.
5	JP9 – data protect function. Bit read as one enables IMS F001 EEROM protection.
6	Latched power-on reset signal. Used in GPIB status reporting; bit reads one if last reset was caused by power-on.
7	SubsystemnotError signal from subsystem port. Bit read as zero indicates that an error is signalled.
8	JP1 – bit 1 (LSB) of GPIB primary address, GPIB_A1. (Bit read as zero means jumper is present)
9	JP2 – bit 2 of GPIB primary address, GPIB_A2.
10	JP3 – bit 3 of GPIB primary address, GPIB_A3.
11	JP4 – bit 4 of GPIB primary address, GPIB_A4.
12	JP5 – bit 5 (MSB) of GPIB primary address, GPIB_A5.
13	JP6 – device capability – see bit 14 and summary immediately below.
14	JP7 – device capability – see bit 13 and summary immediately below.
15	JP8 – GPIB drive selection. Bit read as one requests tristate; bit read as zero requests open-collector.

Bit 14	Bit 13	Capability
0	0	Device, talk and listen
0	1	Device, talk only
1	0	Device, listen only
1	1	Controller, talk and listen

32.6.3 Output port assignments

The I/O ports allow writing of 8 output bits as one byte, at hardware address #5000. The address decoding is such that the same byte be read at hardware addresses #5001, #5002, etc. The usage of the individual bits, beginning with the least significant, is as follows:

Output Bit	Purpose
0	System Control. When set to one, allows the IMS B421 to control the GPIB lines REN and IFC, i.e. to be system controller.
1	Drive selection. When set to zero, the GPIB drivers are open-collector; when set to one, they are tri-state.
2	SubsystemAnalyse. Value written appears as non-inverted control line logic level.
3	SubsystemReset. Value written appears as non-inverted control line logic level.
4	Any value may be written – has no function.
5	Any value may be written – has no function.
6	Any value may be written – has no function.
7	Any value may be written – has no function.

32.6.4 GPIB control register addresses

Hardware address	Read	Write
#5800	Interrupt Status Register 0.	Interrupt Mask Register 0.
#5802	Interrupt Status Register 1.	Interrupt Mask Register 1.
#5804	Address Status Register.	Not used.
#5806	Bus Status Register.	Auxiliary Command Register.
#5808	Not used.	Address Register.
#580A	Not used.	Serial Poll Register.
#580C	Command Pass Through Register.	Parallel Poll Register.
#580E	Data In Register.	Data Out Register.

Table 32.5 GPIB control registers

The GPIB controller has an eight bit wide data bus, but for technical reasons has to be mapped as if it were a sixteen bit wide device. This means that if a word (i.e. 16 bit wide) transfer operation is performed, the value in the eight more significant bits of the word is ignored by the controller in a write, and returns no information to the transputer after a read.

The GPIB controller address decoding is such that the register set reappears repeatedly throughout the range shown in the memory map in successive blocks of sixteen bytes.

32.6.5 Wait states

The transputer is in general capable of much faster data bus cycles than the I/O devices on the IMS B421. Therefore, the supporting logic inserts wait states into cycles involving these devices as appropriate. Programmers requiring accurate knowledge of execution times are referred to the following table:

Hardware address	occam address	Device	Waits	Proc. Clock Cycles (Note 1)
#9000 to 4FFF	#0800 to 67FE	Ext. RAM	0	2 (word)
#5000 to 57FF	#6800 to 6BFE	I/O ports	0	2 (word)
#5800 to 5FFF	#6C00 to 6FFE	GPIBC	3 Read 2 Write	5 (byte) 4 (byte) (Note 2)
#6000 to 7FFF	#7000 to 7FFE	EEROM	5	7 (byte) (Note 3)

Notes:

- 1 Each processor clock lasts 50 nsec. (Clock frequency is 20 MHz.)
- 2 Although the GPIB controller is addressed as a word wide device, it can only make use of the less significant byte of the data bus. It is impossible for the device to transfer data bytes in pairs, even if the transputer is programmed to perform 16-bit operations; the more significant byte of each such transfer cannot convey useful information. Consequently, each byte to be transferred will require the number of clock cycles indicated in the table regardless of the transfer operand size.
- 3 Although it is permissible to program the transputer for 16-bit operations with the EEROM, the IMS B421 control logic will transparently break these down into two 8-bit transfers, each requiring 7 clock cycles.

32.6.6 Internal and external reset pulse generation

The internal power-on reset signal is produced by a TL7705 supply voltage monitor. This is configured to produce a pulse of at least 10 ms. width whenever the power rail drops below 4.5V ($\pm 0.05V$). The device is included both to provide the power-on status information incorporated in the IEEE-488 status reporting model, and to ensure orderly behaviour of the TRAM and any attached subsystems upon application of power.

Note that an external reset pulse applied to the IMS B421 is not modified by the presence of the TL7705, i.e. neither its leading nor trailing edges are delayed at all, at the T222 or at the subsystem pins. The TRAM reset pin and the supply voltage monitor chip are entirely independent sources of resets to the transputer and subsystem.

When the IMS B421 is reset, either by an external pulse or by detection of a rising edge on the power rail, the reset pulse is propagated to the subsystem pins.

32.6.7 EEROM programming

EEROM programming must always observe certain timing constraints, and to achieve maximum speed a particular algorithm must be used. This algorithm is implemented in the IMS F001 software so that the user need not be concerned with its details. The use of the IMS F001 EEROM access commands will ensure that all EEROM operations are performed at the highest speed attainable by the hardware. A full description of the programming process at hardware level is beyond the scope of this document.

IMPORTANT: The programmer must exercise **CAUTION** with respect to the EEROM. The device technology imposes a limit on the number of write cycles which a memory cell may endure; when the cell's limit is exceeded, it will no longer accept and retain the data written. The EEROM devices fitted to the IMS B421 are characterised by their manufacturers to perform correctly after 10,000 write cycles per cell, at a mini-

mum. This figure is an estimate derived from device reliability statistics, and in practice a particular cell may continue to operate correctly for many more than 10,000 write cycles. However, to maintain confidence that the device will retain the desired data, the application should constrain the number of writes to any EEROM location well within the 10,000 cycle limit, over the anticipated maintenance interval of the equipment. As an additional caution, note that inadequately tested user code might unintentionally perform a large number of EEROM writes in a very short time, exceeding the write cycle endurance of some or all cells. The user should take all reasonable steps to avoid this possibility when developing new IMS B421 code.

32.7 Mechanical details

Figure 32.5 indicates the vertical dimensions of a single IMS B421 and figure 32.6 shows the outline drawing of the IMS B421.

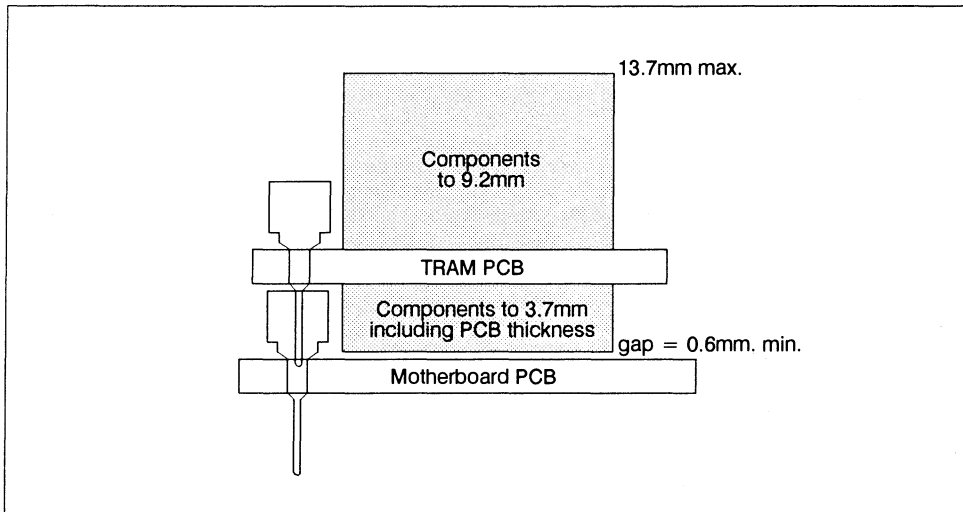


Figure 32.5 IMS B421 height specification

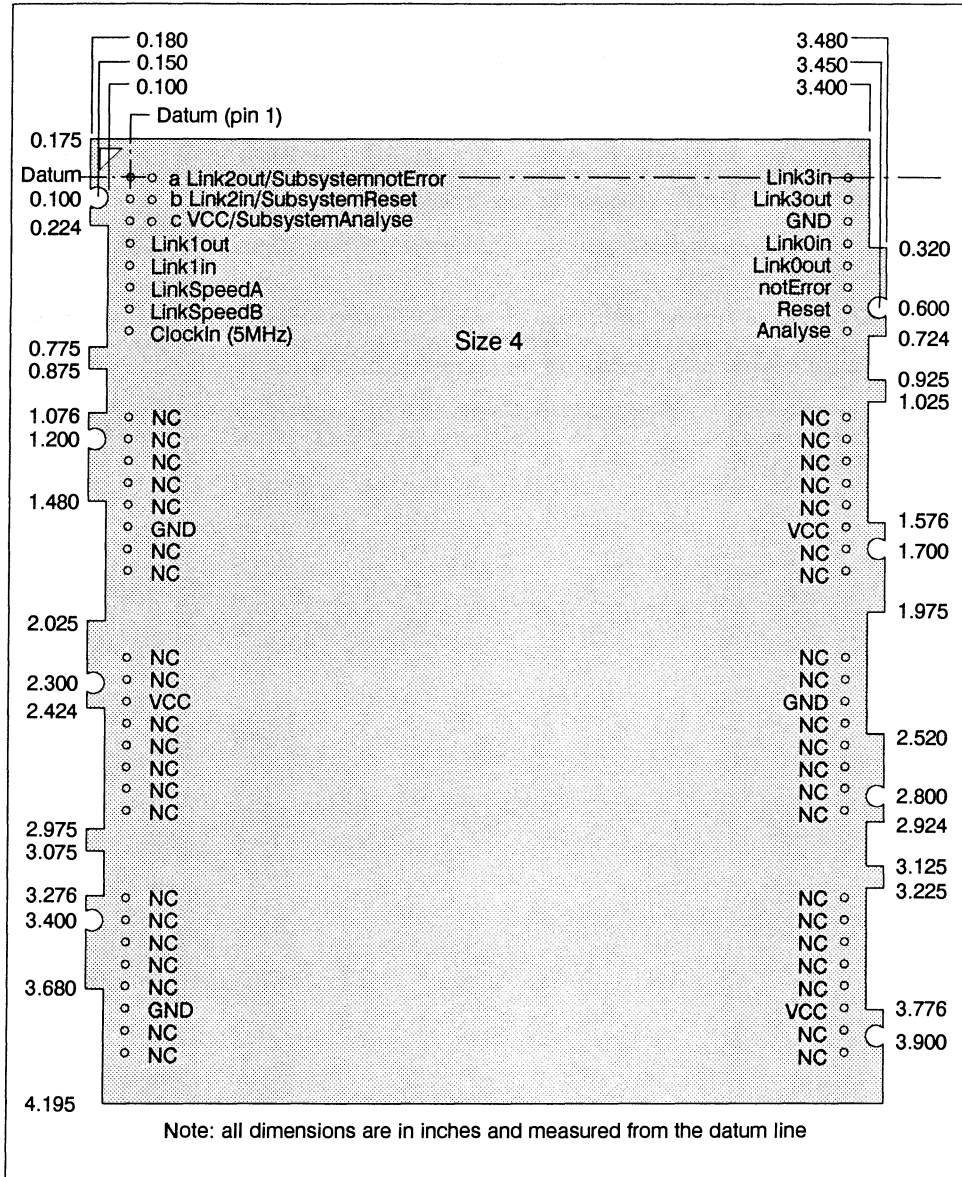


Figure 32.6 IMS B421 PCB profile drawing and pinout

32.8 Installation

Since the IMS B421 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B421 is supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help to protect the TRAM pins during transit. Secondly, they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

If the subsystem signals are required, plug a 3-way header strip into the solder-side sockets (aside pins 1–3) on the IMS B421.

Plug the IMS B421 into the motherboard. Where the IMS B421 is being used with an INMOS motherboard, the silk screened triangle marking pin 1 on the IMS B421 (see figure 32.6) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B421, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B421 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

32.9 Specification

TRAM feature	IMS B421	Units	Notes
Transputer type	IMS T222-20		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	48	KBytes	
Memory Wait States	0		1
Memory cycle time	100	ns	
Subsystem controller	Yes		
Peripheral circuitry	IEEE-488 Interface		
Memory Parity	No		
Size (TRAM Size)	4		
Length	3.66	Inch	
Pitch between pins	3.30	Inch	
Width	4.35	Inch	
Component height above PCB	9.2	mm	
Component height below PCB	3.7	mm	2
Weight	77	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	3
Power supply voltage (VCC)	4.75 -5.25	Volt	
Power consumption	7	W	4

Table 32.6 IMS B421 specification

Notes:

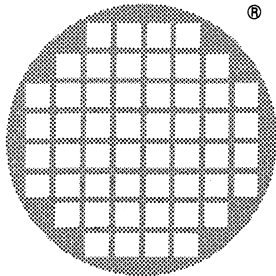
- 1 EEROM and I/O device cycles include more waitstates; see programmer's information for details.
- 2 This dimension includes the thickness of the PCB.
- 3 The figure quoted refers to the ambient air temperature.
- 4 The power consumption is the worst case value obtained when a sample of IMS B421 TRAMs was tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25V.

32.10 Ordering information

Description	Order Number
IMS B421 GPIB TRAM with T222-20	IMS B421-10 *
IMS F001 GPIB library software	IMS F001A-1

Table 32.7 Ordering information

*Includes IMS F001A-1 GPIB library software



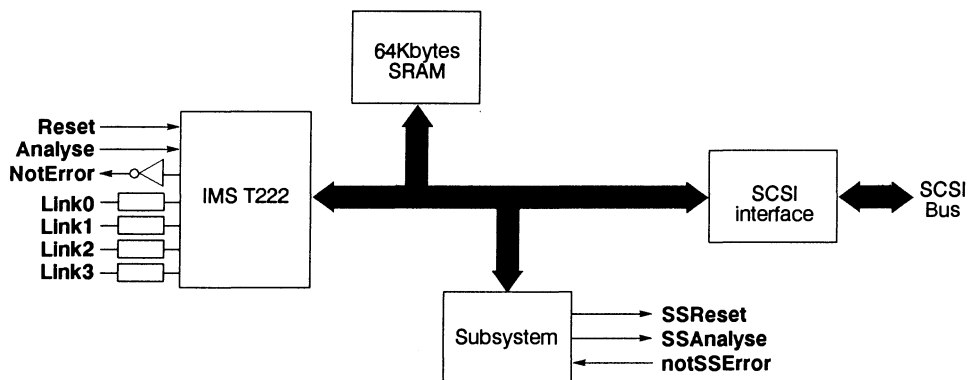
inmos[®]

IMS B422

SCSI TRAM

Size 2

Engineering Data



FEATURES

- IMS T222-20 transputer
- 64 Kbytes of two-cycle memory
- SCSI bus interface (single ended drivers)
- Sustained SCSI transfer rates up to 1.5MBytes/s
- Target and Initiator modes
- On-board, user removable SCSI bus terminators
- Subsystem port
- Size 2 TRAM
- Designed to a published specification (*INMOS Technical Note 29*)
- Supplied with comprehensive board support software (IMS F002)
- Software supports multi-threaded I/O

GENERAL DESCRIPTION

The SCSI TRAM acts as an interface between an INMOS link and the SCSI bus as defined in the ANSI X3.131-1986 standard. It allows transputer systems to connect to winchester disks, optical disks, and other peripherals via the SCSI bus. The SCSI TRAM consists of an IMS T222 16 bit transputer with 64 Kbytes of SRAM for program and data buffers. An intelligent interface device is used to implement the connection to the SCSI bus which allows common sequences to proceed without intervention from the IMS T222. Target and initiator modes are supported allowing use in *hosts* or *peripherals*. On board removable SCSI bus terminators are provided. A standard subsystem port is implemented on the TRAM.

33.1 IMS B422 SCSI TRAM engineering data

33.1.1 Transputer Modules (TRAMs)

The IMS B422 is one of a range of INMOS TRANSputer Modules (TRAMs) and incorporates an IMS T222-20 transputer, 64 Kbytes of static RAM, and an interface to the SCSI bus. TRAMs are subassemblies of transputers, memory and peripheral devices. They interface to each other via INMOS links, have a standard pinout, and come in a range of standard sizes¹. TRAMs allow powerful, flexible, transputer based systems to be produced with the minimum of design effort. The standard TRAM interface signals are described below.

33.1.2 Pin descriptions

Pin	In/out	Function	Pin no.
System services			
VCC, GND		Power supply and return	3, 14
ClockIn	in	5MHz clock signal	8
Reset	in	Transputer reset	10
Analyse	in	Transputer error analysis	9
notError	out	Transputer error indicator (inverted)	11
Links			
LinkIn0-3	in	INMOS serial link inputs to transputer	13, 5, 2, 16
LinkOut0-3	out	INMOS serial link outputs from transputer	12, 4, 1, 15
LinkspeedA, B	in	Transputer link speed selection	6, 7
Subsystem services			
SubSystemReset	out	Subsystem reset	1b
SubSystemAnalyse	out	Subsystem analyse	1c
notSubSystemError	in	Subsystem error indicator	1a

Table 33.1 IMS B422 Pin designations

Notes

- 1 Signal names are prefixed by **not** if they are active low; otherwise they are active high.
- 2 Details of the physical pin locations can be found in figure 33.6.

ClockIn

A 5MHz input clock for the transputer. The transputer synthesises its own high frequency clocks. **ClockIn** should have a stability over time and temperature of 200ppm. **ClockIn** edges should be monotonic within the range 0.8V to 2.0V with a rise/fall time of less than 8ns.

LinkOut0-3

Transputer link output signals. These outputs are intended to drive into transmission lines with a characteristic impedance of 100Ω. They can be connected directly to the **LinkIn** pins of other transputers or TRAMs.

¹ Further details of the TRAM/motherboard philosophy and the full electrical and mechanical specification of TRAMs can be found in INMOS Technical Note 29: *Dual-In-Line Transputer Modules (TRAMs)* and INMOS Technical Note 49: *Module Motherboard Architecture*. The *Transputer Reference Manual* may also be of use. These are available as separate publications from INMOS.

LinkIn0-3

Transputer link input signals. These are the link inputs of the transputer on the IMS B422. Each input has a 10kΩ resistor to **GND** to establish the idle state, and a diode to **VCC** as protection against ESD. They can be connected directly to the **LinkOut** pins of other transputers or TRAMs.

LinkSpeedA, LinkSpeedB

These select the speeds of **Link0** and **Link1,2,3** respectively. Table 33.2 shows the possible combinations.

LinkSpeedA	LinkSpeedB	Link0	Link1,2,3
0	0	10 Mbits/s	10 Mbits/s
0	1	10 Mbits/s	20 Mbits/s
1	0	20 Mbits/s	10 Mbits/s
1	1	20 Mbits/s	20 Mbits/s

Table 33.2 Link speed selection

Reset

Resets the transputer, and other circuitry. **Reset** should be asserted for a minimum of 100ms. After **Reset** is deasserted a further 100ms should elapse before communication is attempted on any link. After this time, the transputer on this TRAM is ready to accept a boot packet on any of its links.

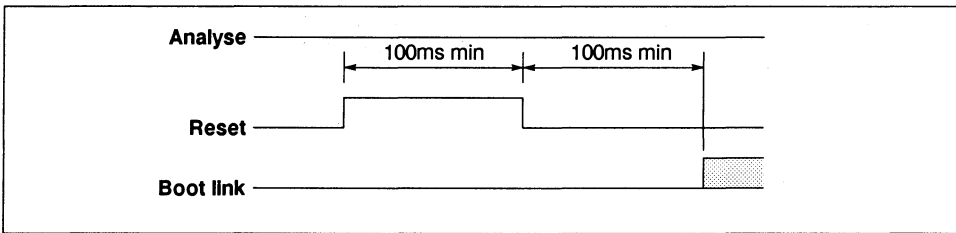


Figure 33.1 Reset timing

Analyse

This is used, in conjunction with **Reset**, to stop the transputer. It allows internal state to be examined so that the cause of an error may be determined. **Reset** and **Analyse** are used as shown in figure 33.2. A processor in analyse mode can be interrogated on any of its links.

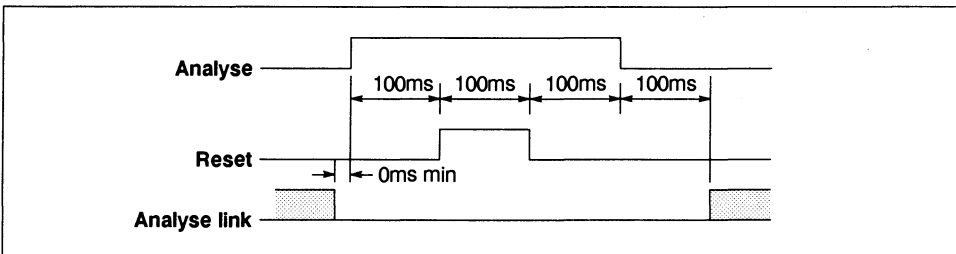


Figure 33.2 Analyse timing

notError

An open collector output which is pulled low when the transputer asserts its Error pin. **notError** should be pulled high by a 10kΩ resistor to **VCC**. Up to 10 **notError** signals can be wired together. The combined error signal will be low when any of the contributing signals is low.

33.1.3 A Brief Description of the Small Computer System Interface.

The Small Computer System Interface (SCSI), as described in reference documents 1 and 2, is a local I/O bus that can be operated at data rates upto 4 megabytes per second depending upon circuit implementation choices. The primary objective of the interface is to provide host computers with device independence within a class of devices. Thus different disk drives, tape drives, printers, and communication devices can be added to the host computer(s) without requiring modifications to generic system hardware or software.

Transfer of data across the bus is started by an *initiator* selecting a SCSI device and sending a command to it. The selected SCSI device is referred to as a *target*. From this point on the target controls transfers on the bus to transfer the data requested by the command. A message passing system is used between the initiator and the target to control the interface. Provision is made for the addition of nongeneric features and functions through vendor unique fields and codes.

Communication on the SCSI bus is allowed between only two SCSI devices at any given time. There is a maximum limit of eight SCSI devices on a SCSI bus.

33.1.4 SCSI Capabilities

The IMS B422 allows transputer based systems to communicate with up to seven SCSI devices on a single bus, providing these systems with a way of interfacing to a large range of peripherals. This range includes disk drives, both magnetic and optical, tape drives, printers, CD-ROM, scanners, communication devices, and other computer systems. Data transfer rates up to 1.5 Mbytes/s can be supported on the SCSI bus. Physically the IMS B422 implements a connection to a single ended driver type of SCSI bus. It connects to the bus by a 50 way header on one edge of the TRAM.

The IMS B422 in conjunction with the IMS F002 board support software has the following SCSI features.

- Single ended drivers
- Terminator power supplied to the cable
- Parity generated, detection of parity optional
- 'Soft' or 'hard' SCSI bus reset options
- Target or Initiator modes
- Synchronous data transfers supported
- Maximum REQ/ACK offset 15
- Minimum transfer period 200 ns
- Sustained data transfer rates up to 1.5 Mbytes/s
- Software supports multi-threaded I/O
- Requests from multiple transputers can be serviced
- Data can be transferred over multiple links to increase the data rate to/from the SCSI TRAM.
- Directly supports the issuing of commands of the Common Command Set (CCS) through procedural interfaces.

33.1.5 Connecting the IMS B422 to a SCSI bus

The SCSI bus is implemented as a cable containing 50 wires which connects together the SCSI devices on the bus in a daisy-chain arrangement. At the two ends of the cable the signal lines must be terminated

either with termination resistors in a SCSI device, or with external terminators. The SCSI standard gives details of the cable types and connectors that should be used to implement the bus.

The IMS B422 SCSI connector is a 50 way dual row shrouded header with a polarising slot, designed to mate with a 50 thou pitch ribbon cable IDC socket. Use of polarised connectors is recommended.

In view of the mechanical characteristics of TRAMs and TRAM motherboards it is not recommended that an unsupported SCSI bus cable is taken directly to the connector on the IMS B422. The recommended method of connecting the SCSI bus to a system containing a IMS B422 is to plug the external cable into a connector mounted on the case of the system, which is connected via a short cable to the IMS B422.

In the SCSI specification there are constraints placed on the geometry of the SCSI bus. To meet these constraints with the IMS B422 requires some attention to be paid to the method of connecting the TRAM to the SCSI bus.

There are two situations :

- 1 The IMS B422 is connected to the end of the bus.
- 2 The IMS B422 is connected to the middle of the bus.

In case 1 the bus can be plugged into a connector on the outside of the system containing the IMS B422 with the terminator resistor packs on the TRAM plugged in.

In case 2 the bus must be cabled from the outside of the system, to the IMS B422, and back to another connector on the outside of the system. The terminator resistor packs must also be removed from the IMS B422. The bus can not be just passed through the external connector as this would leave a *stub* length of bus connected to the main bus. The SCSI standard specifies that no stubs are allowed to be greater than 10 cm in length on a single ended SCSI bus cable, which such an arrangement would almost certainly exceed.

This second arrangement of cabling allows for the situations in both case 1 and 2 if the terminator resistor packs on the IMS B422 are removed and external terminators are used. This allows an end user to cable a SCSI bus between a number of devices without having to access the SCSI TRAM.

Terminator power is supplied to the SCSI bus from the IMS B422 via a fuse and a diode. The diode prevents power being drawn from the SCSI bus if another device on the bus if also providing terminator power at a higher voltage. A fuse has been included to protect the IMS B422 and the power supply it is connected to, in the event of the TERMPWR pin on P1 being shorted to ground.

Figure 33.7 shows the position and orientation of the SCSI bus connection, P1 on the IMS B422. The pinout of P1 is shown in figure 7.

33.1.6 IMS B422 hardware

The design of the IMS B422 complies with the SCSI standard, reference 1. An intelligent SCSI interface controller IC is used to implement the interface to the SCSI bus. This device handles the low level protocols involved in accessing the bus and transferring information, relieving the IMS T222 of these time critical tasks, speeding up data transfer, and ensuring compliance with the timings in the SCSI specification. The IMS B422 can be operated in target or initiator modes allowing the TRAM to be used either in 'Hosts', peripherals, or for host to host communication as defined in the SCSI-2 specification, reference 2.

A 16 byte data FIFO is built into the controller and the hardware allows data transfers to and from this FIFO to be carried out using a block transfer mode, ensuring a fast data transfer rate between a SCSI device and memory. Parity generation and checking by the controller can be enabled. The controller also supports some of the features of SCSI-2, allowing the messages necessary for command queuing at the drive, and group 2 commands, to be passed between the initiator and the target.

33.1.7 Memory map

The IMS T222 on the IMS B422 can access a 64K address range. The internal RAM of the IMS T222 is mapped into the first 4 Kbytes of the address space with the remainder of the address space mapped to 59.75 Kbytes of external static RAM and the memory mapped I/O.

Address	Function
#8000 – #8FFF	Internal memory
#9000 – #7F00	External 2 cycle memory
#7F00 – #7F1F	SCSI controller registers byte wide on even bytes
#7F20 – #7F3F	SCSI controller registers byte wide on even bytes repeated
#7F40 – #7F5F	SCSI controller FIFO register word wide 8 times
#7F60 – #7F7F	SCSI controller FIFO register word wide 8 times repeated
#7F80 – #7F83	Subsystem port registers
#7F84 – #7FBF	Subsystem port registers repeated
#7FC0 – #7FC1	DRQ flag register
#7FC2 – #7FFF	DRQ flag register repeated

Table 33.3 Memory map of the IMS B422

Table 33.3 shows the address map of the IMS B422 (the '#' sign indicates a hexadecimal number). Addresses range from #8000 through #0000 to #7FFF. The internal RAM on the IMS T222 occupies the first 4Kbytes of address space. The memory mapped I/O occupies the top 256 bytes of memory. The internal RAM on the IMS T222 has a 50ns access cycle and the external SRAM has a 100ns access cycle. Note partial decoding is used for the memory mapped I/O address decoding resulting in the registers appearing at more than one address.

33.1.8 IMS F002 board support software

The IMS F002 SCSI support software package provides a low level interface between a user application and the IMS B422. The interface presented to the application program is intended to abstract hardware implementation details from the caller, thus mimimising the impact of any future hardware upgrades.

33.1.9 Structure

IMS F002 consists of 4 major components :-

- 1 IMS B422 device driver.
- 2 Initialisation interface.
- 3 Initiator mode interfaces (Host computer interface).
- 4 Target mode interfaces (Peripheral operation interface).

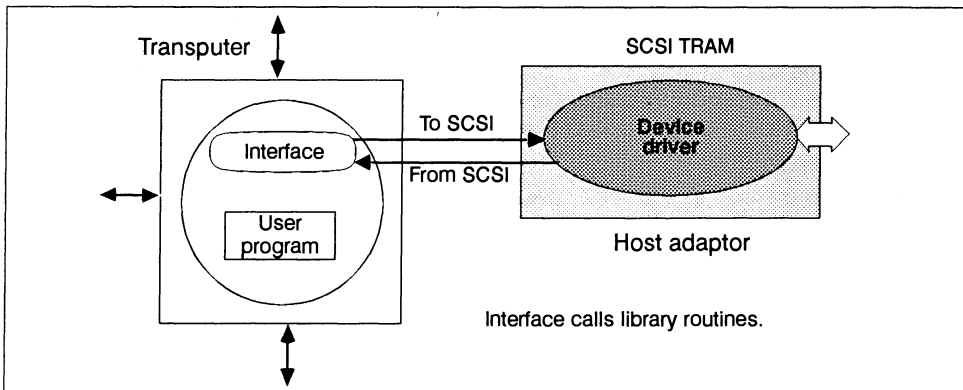


Figure 33.3 Software Elements

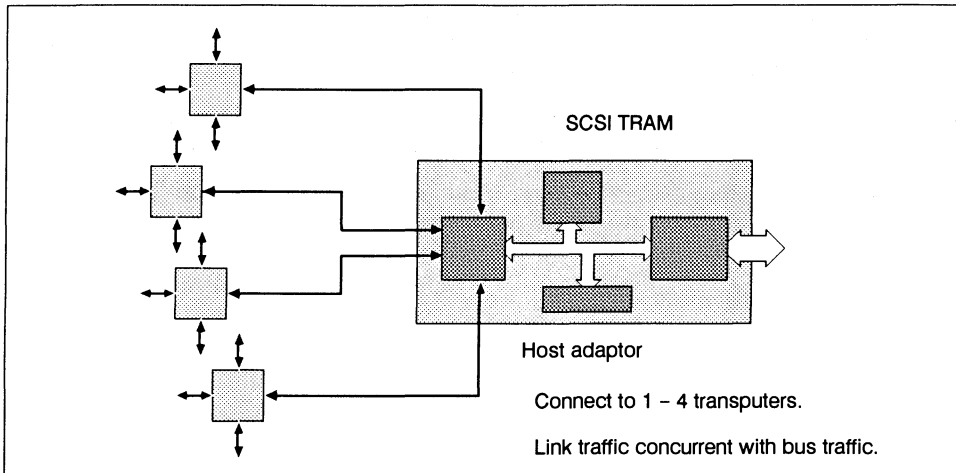


Figure 33.4 Hardware Configuration

33.1.10 IMS B422 Device Driver.

The IMS B422 device driver resides on IMS B422 and is responsible for :-

- 1 Providing the facilities of a Host Adaptor Device Driver for initiator mode interfaces.
- 2 Receiving SCSI requests presented by the SCSI bus and forwarding on the received requests to transputers connected to the 4 transputer links of IMS B422 when in target mode.
- 3 Performing diagnostics upon the hardware of the SCSI interface circuitry and data buffer areas used for SCSI bus data transfer on the IMS B422.

For the purpose of communicating to connected transputers, the IMS B422 device driver uses the channel protocols TO.SCSI (input channel) and FROM.SCSI (output channel). The use of these protocols permits SCSI configuration information, SCSI commands and associated data packets to be transferred across the same transputer link. Optionally, data throughput may be increased by specifying any one of the other unused transputer links of IMS B422 to be an additional data only link to be used in tandem with the currently accessed link. It is not necessary for the application to directly interface at link protocol level.

33.1.11 Initialisation Interface.

The Initialisation interface permits the user to define operating characteristics of the IMS B422. It also permits self test to be performed.

33.1.12 Initiator Mode Interface.

The Initiator mode interfaces provide transputer applications executing upon transputers connected to IMS B422 the ability to access SCSI peripherals via IMS B422.

A generic driver (Host Adaptor Device Driver Interface HADDIF), is provided that accepts SCSI Command sequences and appropriate data buffer areas. The supplied SCSI Command sequence is issued to the IMS B422 device driver for execution by the specified SCSI target device. HADDIF then provides the target device via IMS B422, the ability to access the supplied data areas on the connected transputer in order to provide the initiator requested service. Whilst HADDIF is executing, the CPU resources for SCSI I/O of HADDIF's transputer are kept to an absolute minimum, thereby giving maximum CPU resources to the user's application.

Specific support is provided for SCSI commands that are members of the Common Command Set. These SCSI commands can be found on the majority of Direct Access SCSI Winchester disk drives. For each member of the Common Command Set, two interfaces are provided.

The primary interface form accepts non byte packed parameters, transparently builds a byte packed SCSI command sequence and issues it to HADDIF for execution by a specified target device.

A secondary interface form also accepts non byte packed parameters and returns a built byte packed SCSI command sequence. No access is made to IMS B422 by this form of interface. The user must specifically make a call to HADDIF in order to execute the built SCSI command sequence.

Example of primary interface form :-

(occam):

```
PROC SCSI.Read.10( CHAN OF TO.SCSI      TO.SCSI.HA,
                  CHAN OF FROM.SCSI    FROM.SCSI.HA,
                  VAL BYTE              Target.Id,
                  VAL BYTE              LUN,
                  VAL BYTE              dps,
                  VAL BYTE              tpa,
                  VAL BYTE              reladr,
                  VAL INT32             Logical.Block.Address,
                  VAL INT32             Number.of.Blocks,
                  VAL INT32             Block.Size,
                  VAL BYTE              Control.Byte,
                  []BYTE                Rx.Data,
                  BYTE                  Msg.Length,
                  []BYTE                Message,
                  BYTE                  SCSI.Status,
                  INT16                  Execution.Status)
```

(C):

```
int scsi_read_10( channel      *to_scsi_ha,
                  channel      *from_scsi_ha,
                  char          target_id,
                  char          lun,
                  char          dpo,
                  char          fua,
                  char          reladr,
                  int           logical_block_address,
                  int           number_of_blocks,
                  int           block_size,
                  char          *control_byte,
                  char          rx_data[],
                  char          *msg_length,
                  char          message[],
                  char          *scsi_status)
```

A call to the above procedure will build and issue an SCSI Read10 command to the specified Target.ID and Logical Unit (LUN) via HADDIF, IMS B422 Device Driver and IMS B422 SCSI TRAM. Data read from the disk starting at Logical.Block.Address for Number.of.Blocks will be written into the supplied data area Rx.Data. The total number of bytes read from the disk will be Number.of.Blocks * Block.Size bytes.

If successful, an Execution.Status of SCSI.E.Good will be returned.

If Execution.Status is returned as SCSI.E.Bad.SCSI.Status, then the caller should inspect SCSI.Status, Msg.Length and Message, which are directly returned by the target peripheral device. It will probably be necessary to issue a Request.Sense command to the target peripheral device subsequently to this error condition, in order to ascertain the precise nature of the error condition and to clear the error condition.

Other returned values of Execution.Status are specific to the operation of IMS B422 and its device driver.

33.1.13 Target Mode Interface.

The Target mode interfaces provide transputer systems acting as peripheral devices, the ability to receive and execute SCSI Command Sequences supplied by initiator devices on the SCSI Bus. Interfaces are provided to initially receive a SCSI Command and for each SCSI Bus phase.

The use of target mode interfaces requires that the caller has a thorough working knowledge of the SCSI specification and the operation of peripheral devices

33.1.14 Software Distribution

The IMS B422 support software, IMS F002, will be supplied as binary libraries compatible with the INMOS OCCAM and C toolset products. Include files containing the protocol and constant definitions required to interface to the libraries will also be supplied. These will be supported by user documentation.

Distribution media will be 360K 48 TPI 5.25" and 720K 135 TPI 3.5" IBM PC format disks.

33.1.15 Mechanical details

Figure 33.5 gives the vertical dimensions of an IMS B422 and figure 33.6 is an outline drawing of the IMS B422. When the IMS B422 is mounted as shown in figure 33.5, the motherboard and IMS B422 together will occupy two slots in a 0.8 inch pitch card cage.

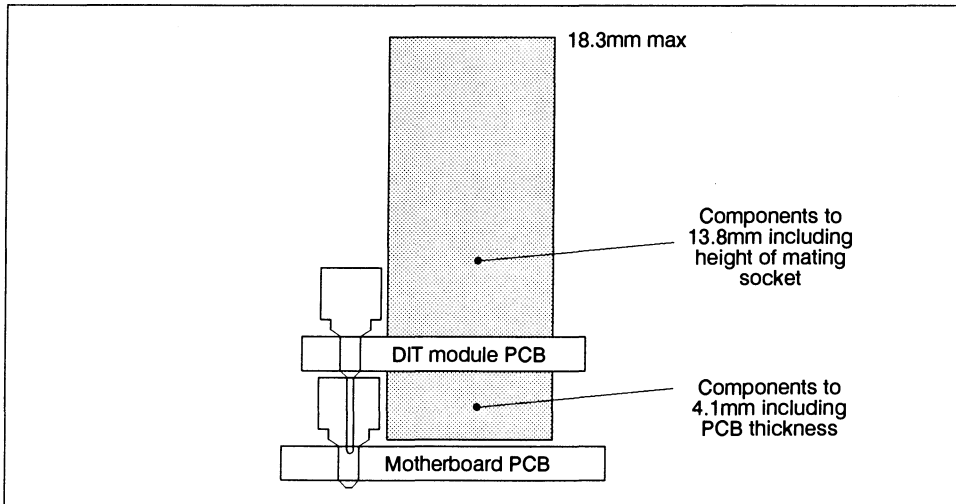


Figure 33.5 IMS B422 height specification

33.1.16 Installation

Since the IMS B422 contains CMOS components, all normal precautions to prevent static damage should be taken.

The IMS B422 will be supplied with spacer pin strips attached to the TRAM pins on the underside of the board. These spacers perform two functions. Firstly, they help protect the TRAM pins during transit. Secondly they can be used to space the TRAMs off the motherboard. If there are no components mounted on the motherboard TRAM slot, then the spacer strips should be removed before the TRAM is inserted.

If the subsystem signals are to be used, plug a 3-way header strip into the solder side sockets on the IMS B422.

Plug the IMS B422 into the motherboard. Where the IMS B422 is being used with an INMOS motherboard, the yellow triangle marking pin 1 on the IMS B422 (see figure 33.6) should be aligned with the silk screened triangle that appears in the corner of the appropriate TRAM slot.

Should it be necessary to unplug the IMS B422, it is advised that it is gently levered out while keeping it as flat as possible. As soon as the IMS B422 is removed, the spacer pin strips should be refitted to the TRAM to protect the pins.

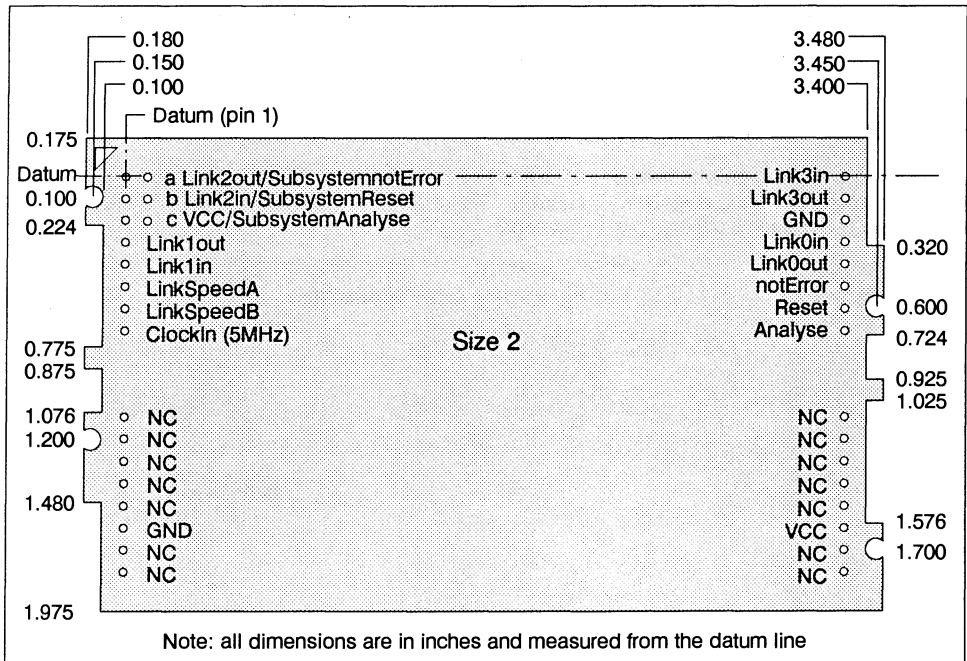


Figure 33.6 IMS B422 outline drawing (All dimensions in inches)

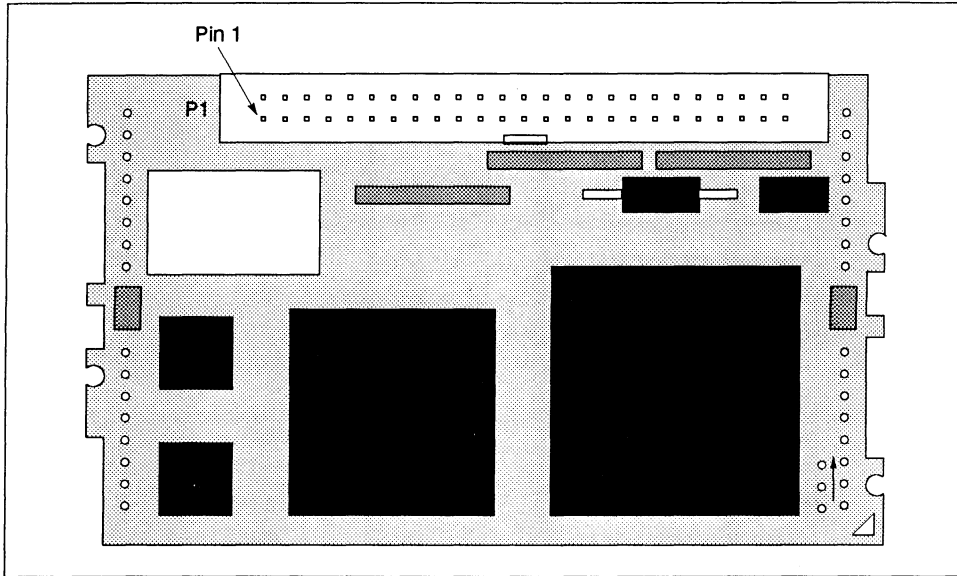


Figure 33.7 Top view of the IMS B422

TOP VIEW

	1	2	
	□	□	-DB(0)
GROUND	□	□	-DB(1)
GROUND	□	□	-DB(2)
GROUND	□	□	-DB(3)
GROUND	□	□	-DB(4)
GROUND	□	□	-DB(5)
GROUND	□	□	-DB(6)
GROUND	□	□	-DB(7)
GROUND	□	□	-DB(P)
GROUND	□	□	GROUND
GROUND	□	□	GROUND
GROUND	□	□	GROUND
OPEN	□	□	TERMPWR
GROUND	□	□	GROUND
GROUND	□	□	GROUND
GROUND	□	□	-ATN
GROUND	□	□	GROUND
GROUND	□	□	-BSY
GROUND	□	□	-ACK
GROUND	□	□	-RST
GROUND	□	□	-MSG
GROUND	□	□	-SEL
GROUND	□	□	-C/D
GROUND	□	□	-REQ
GROUND	□	□	-I/O
	49	50	

Figure 33.8 Pinout of the SCSI bus connector P1

33.1.17 Specification

TRAM feature		Unit	Notes
Transputer type	IMS T222-20		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	64	Kbyte	
Memory "wait states"	0		
Memory cycle time	100	ns	
Subsystem controller	yes		
Peripheral circuitry	SCSI Interface		
Memory Parity	no		
Size (TRAM size)	2		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	2.15	inch	
Component height above PCB	13.9	mm	1
Component height below PCB	4.1	mm	2
Weight	50	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	3
Power supply voltage (VCC)	4.75-5.25	V	
Power consumption	5	W	4

Table 33.4 IMS B422 specification

Notes

- 1 This dimension is larger than is normally stated for TRAMs because of the requirement to connect to SCSI. The quoted height includes the additional height of a 50 way IDC socket, minus strain relief, plugged into the SCSI bus connector.
- 2 This dimension includes the thickness of the PCB.
- 3 The figure quoted refers to the ambient air temperature.
- 4 The power consumption value was obtained from calculations of the worst case figure.

33.1.18 References

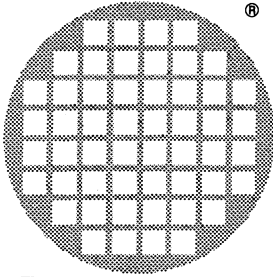
- 1 *Small computer systems interface (SCSI)*, ANSI standard X3.131-1986
- 2 *Small computer systems interface - 2 (SCSI-2)*, ANSI draft standard X3.131-198X

33.1.19 Ordering Information

Description	Order number
IMS B422 SCSI TRAM with IMS T222-20	IMS B422-10*
IMS F002 Support Software	IMS F002B-1

* Includes IMS F002B-1, supplied on 5¼in and 3½in DOS formatted diskettes

Table 33.5 Ordering information



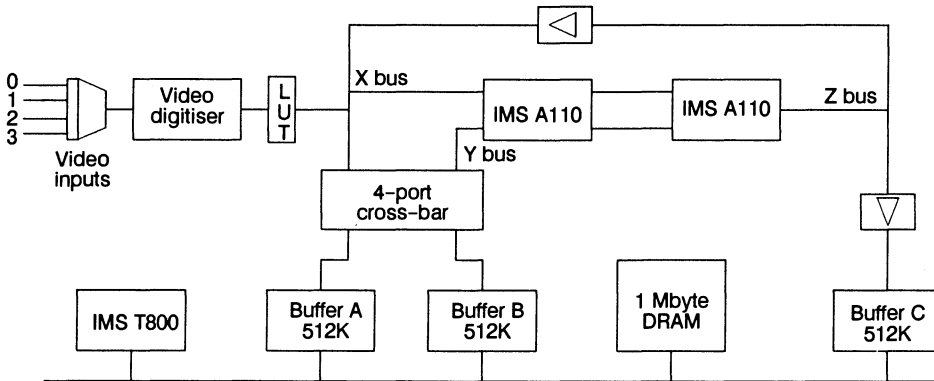
inmos®

IMS B429

Video Image Processing TRAM (VIP)

Advance Information

The information in this section is subject to change



FEATURES

- Real time (30f/s) monochrome image capture and 2d image filtering.
- Can capture from PAL, NTSC, and other image formats.
- Programmable capture resolution and sampling rate: eg 512×512
- Can be used stand-alone for 2d image filtering operations.
- Convolution kernels up to 7×6 or 14×3.
- Images can be recirculated through the IMS A110s for multiple filtering operations (with a corresponding reduction in frame processing rate).
- Supports processing of full frames and image sub-regions.
- Decimation and interpolation in ratios of 2:1 and 4:1.
- Four, multiplexed input channels.
- Can be used for colour processing of static images.
- Input look-up table.
- IMS T800, with 1Mbyte dedicated program/data store, provides link IO and general purpose processing (eg feature extraction).
- Size 6 Transputer Module (TRAM).
- Supported by a comprehensive software library
- Designed to a published specification
(INMOS Technical Note 29)

SGS-THOMSON
MICROELECTRONICS

INMOS is a member of the SGS-THOMSON Microelectronics group

8.1 Introduction

The IMS B429 is a TRAM for real-time image capture and front-end image processing. Front-end image processing is provided by two IMS A110 image and signal processing sub-system devices, which can perform 2d convolution filtering operations with filter kernels of various sizes up to 7×6 , 14×3 , or 42×1 .

8.2 Architecture

The architecture of the IMS B429 is illustrated by figure 1. It can be partitioned into four main blocks: video input, IMS A110s, frame buffers, and processor. These are connected to a set of data-paths.

There are three main video data paths: Image Input (X) Bus, A110 Input (Y) Bus, A110 Output (Z) Bus. The Z bus can be gated onto the X bus to allow an image to be passed more than once through the A110s. The Y bus is the main input to the IMS A110s and is connected to the A110's PSR input port: it can source data from either buffer A or buffer B. The X bus, as well as being the input and feedback bus, is also connected to the CASCADE input port of the IMS A110s: this allows two images, stored in buffers A and B, to be added together during processing.

There are three frame buffers (A,B,C): each of these consists of 512kbytes of dual-port DRAM (VRAM), and each can hold two images of 512×512 pixels. The buffers are connected to the various data paths (X, Y, and Z busses) as shown in the diagram. The A and B buffers are used for intermediate image storage, and can also be used for deinterlacing images before they are processed by the IMS A110s. Buffer C is used to receive images which do not require further processing by the A110s, and is connected directly to the Z bus. The A and B buffers can be used in ping-pong fashion, and are connected to the X and Y busses by a 4-port crossbar switch.

There are two IMS A110s which are connected so that they can be used either in vertical cascade mode, to provide a 7×6 convolution kernel; or in horizontal cascade mode, to provide a 14×3 convolution kernel. Smaller convolution kernels are supported by programming the appropriate coefficients in these larger kernels to be zero.

The CPU, an IMS T800, can access all of the frame buffers through their random access ports. This allows images to be loaded directly into buffers A or B for processing by the IMS A110s. The processor is responsible for all data movements on the board: in particular, transferring data between the bulk VRAM and the VRAM serial ports. The CPU also has 1Mbyte of memory for program storage and holding set-up data for the IMS A110s etc.

When an input video signal is available, all timing for data capture and data movement operations can be derived from it. In the absence of an input signal, an on-board clock and a software driven GO signal are used to drive data through the IMS A110s. This allows the board to be used stand-alone to perform filtering functions using the IMS A110s.

The video input structure consists of: an input multiplexer, anti-aliasing filter, ADC, and LUT. All input channels share the same LUT. Although there are multiple input channels, the board is mainly intended for monochrome capture and processing work, as it does not, on its own, provided co-sited RGB or YUV (YIQ) samples. It can be used to do non-real-time colour capture and processing of static images, by connecting the camera RGB outputs to separate input channels and capturing successive frames from the different channels. Real-time colour work is supported by the ability to synchronise three boards.

34.3 Supported operations

The following processing operations are supported (where * represents the convolution operation and F is the filter kernel)

3 $C = A * F$; B = Input; (either operation may be omitted)

4 $C = B * F$; A = Input; (either operation may be omitted)

5 $B = A * F$;

6 $A = B * F$;

7 $C = A + (B * F)$;

8 $C = B + (A * F)$;

When running under processor-generated timing, all of these operations can be performed on whole images, or on sub-regions of stored images. When running under timing derived from a video signal, full frames only may be processed. Since the timing source is software selectable, full frame capture and sub-region processing may be performed consecutively, though not simultaneously. Full frames may be captured and processed simultaneously.

34.3.1 Decimation and interpolation

The IMS B429 datapath can be configured to perform interpolation or decimation in conjunction with a filtering operation. Decimation and interpolation can both be performed in ratios of either 2:1 or 4:1.

In a decimation operation, the IMS A110s should be programmed with filter coefficients suitable for generating correct-valued samples in the resulting decimated image. The output samples of a (decimated) image are stored consecutively in memory: there are no dummy, or non-useful, values stored.

In an interpolation operation, the data path inserts 0-valued samples between the source pixels prior to filtering: the user does not need to insert dummy or repeated values in the source image. The IMS A110s must be programmed with suitable filter coefficients to generate correct values for the interpolated samples in the output image.

34.4 Video input circuit

The IMS B429 has four identical video inputs. They are terminated with 75Ω, and are ac coupled. They are then clamped to a black reference level during the horizontal back porch.

The video input amplifier has a selectable gain to allow maximum use of the dynamic range of the ADC with both 0.7V and 1.0V video signal amplitudes. Gain is set by a single jumper (J1) which should be inserted for 0.7V inputs and removed for 1.0V inputs.

The chrominance (chroma) signals present in composite colour video signals must be suppressed before sampling. Two ceramic filters are fitted, one to filter out NTSC chroma signals, the other to filter out PAL chroma signals. These filters are enabled by fitting either J2 or J3. Both jumpers may be removed for better signal fidelity if there is no chroma signal present in the input.

A low-pass anti-aliasing filter is fitted.

34.4.1 Input look-up table (LUT)

All input samples pass through the input look-up table. The LUT is of 256 locations each containing an 8-bit value. LUT contents are programmed by the IMS T800. Input samples address the LUT and the addressed value is output to the XBUS. This is useful for modifying the intensity distribution of an image prior to processing it through the IMS A110s. If this is not desired, you should program each LUT location with its address so that it becomes transparent.

34.5 Memory map

Feature	Address
DRAM (1Mb)	0x80000000 - 0x800FFFFFF
Image Buffer A	0x80100000 - 0x8017FFFF
Image Buffer B	0x80200000 - 0x8027FFFF
Image Buffer C	0x80300000 - 0x8037FFFF
Transfer Region A	0x80180000 - 0x801FFFFFF
Transfer Region B	0x80280000 - 0x802FFFFFF
Transfer Region C	0x80380000 - 0x803FFFFFF
SubSystem Reset	0x00000000
SubSystem Analyse	0x00000004
A110 reset	0x00000010
master/slave	0x00000014
clock source	0x00000018
Y BUS source	0x0000001C
X BUS source bit0	0x00000020
X BUS source bit1	0x00000024
Field Sync	0x00000028
Odd/Even Field	0x0000002C
Video Digitiser	0x00080000
IMS A110 A	0x00100000 - 0x001007FC
IMS A110 B	0x00200000 - 0x002007FC
GO/PLL multiplier	0x00300000
Width Register	0x00300004
Clock Speed Register	0x00300008
X Delay Register	0x0030000C

Table 34.1 IMS B429 Memory Map

34.6 Specification

TRAM feature		Unit	Notes
Transputer type	IMS T800-20		
Number of INMOS serial links	4		
Amount of DRAM	1	Mbyte	
DRAM wait states	1		
Memory cycle time	200	ns	
Subsystem controller	Yes		
Peripheral circuitry	2 IMS A110 DSP		
TRAM size	6		
Length	3.66	inch	
Width	6.55	inch	
Pitch between pins	3.30	inch	
Component height above PCB		mm	1
Component height below PCB		mm	1
Weight	188	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	
Power supply voltage (Vcc)	4.75-5.25	Volt	
Power consumption (Max)		W	1
Power consumption (Typical)	6.25	W	

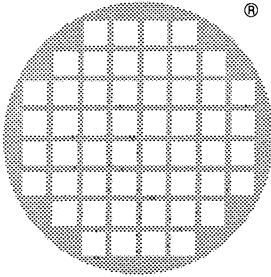
Table 34.2 IMS B429 specification

Notes:

- 1 Not yet specified

34.7 Ordering information

Please contact your local SGS-Thomson sales office for details.

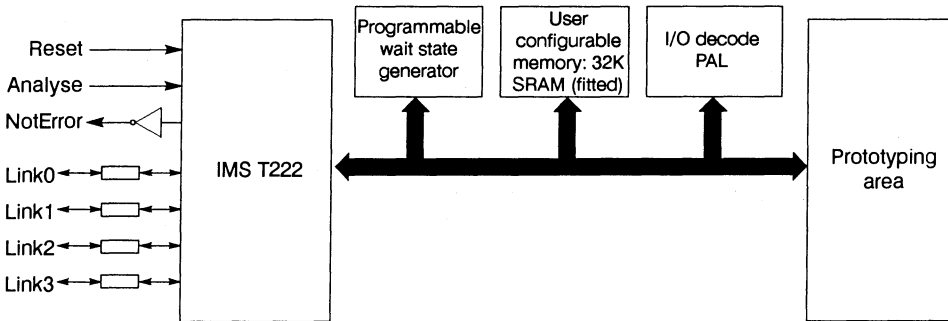


inmos®

IMS B430

Prototyping TRAM

Product overview



FEATURES

- IMS T222 16-bit Transputer
- User programmable Wait State Generator
- Two JEDEC user configurable memory sockets with 32Kbyte of SRAM fitted as standard
- Two memory mapped I/O control signals
- Large through-hole matrix prototyping area with all relevant signals in close proximity
- Size 4 TRAM
- Communicates via 4 INMOS links
- Designed to a published specification (*INMOS Technical Note 29*)

GENERAL DESCRIPTION

The IMS B430 combines a minimal transputer system with a general purpose prototyping area. The product supports feasibility investigations, software development, demonstrations, and other activities requiring construction of a small number of units to a specific design. Inclusion of a functional transputer system speeds up the prototyping process significantly. The user is relieved of the tasks of defining, constructing, and debugging such a system, and need only be concerned with the peripheral hardware specific to the intended application.

35.1 IMS B430 Prototyping TRAM product overview

Printed Circuit Board

The board itself is of high quality four layer construction. All holes are through-plated. The two inner layers carry Power and Ground planes to all devices, including the prototyping area. This form of supply distribution supports reliable operation of high speed digital devices such as transputers, minimising problems of ground-bounce and noise. The two outer layers carry all signal tracks, and there are NO blind or buried vias. Therefore, the user can gain access to all signal connections.

Onboard transputer system

The IMS B430 TRAM has an IMS T222-20 transputer with 4K bytes of fast internal RAM. There is also a socket for a 20-pin Programmable Logic Device (PLD), and two 'JEDEC' sockets for 28-pin memory devices. The transputer itself is socketed, so that it may be upgraded at some future date (e.g. to an IMS T225), or replaced in case of damage.

All the transputer signals - address bus, data bus, etc. - are brought to pads near the prototyping area. These provide convenient points for running signal wires to the prototyping devices, and for the attachment of test probe points for signal monitoring. Where possible, the pads are labelled with signal names (although in some cases the component density prevents this).

Prototyping area

This is made up of 1.0 mm (0.040 in.) through-plated holes in a 2.54 mm (0.1 in.) matrix. These holes will accept the leads of commonly available components, sockets, wire-wrapping terminals, etc. There are 16 columns of holes across the 1.6 in. width of the area, and 43 rows of holes along the 4.3 in. length of the area.

35.1.1 Specification

TRAM feature	IMS B430	Units	Notes
Transputer type	IMS T222-20		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of RAM	4-64	Kbyte	
Memory Wait States	Programmable 0-3		
Memory cycle time	100	ns	
Subsystem controller	No		
Peripheral circuitry	Prototype area		
Memory Parity	No		
Size (TRAM Size)	4		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	4.35	inch	
Component height above PCB	9.2	mm	
Component height below PCB	3.7	mm	1
Weight	-	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	2
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	1	Watt	3

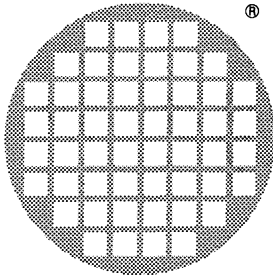
Notes

- 1 This dimension includes the thickness of the PCB.
- 2 The figure quoted refers to the ambient air temperature.
- 3 The power consumption is the worst case value obtained when a sample of IMS B430 TRAMs was tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25V.

35.1.2 Ordering Information

Description	Order number
IMS B430 Prototyping TRAM	IMS B430-10

Table 35.1 Ordering information



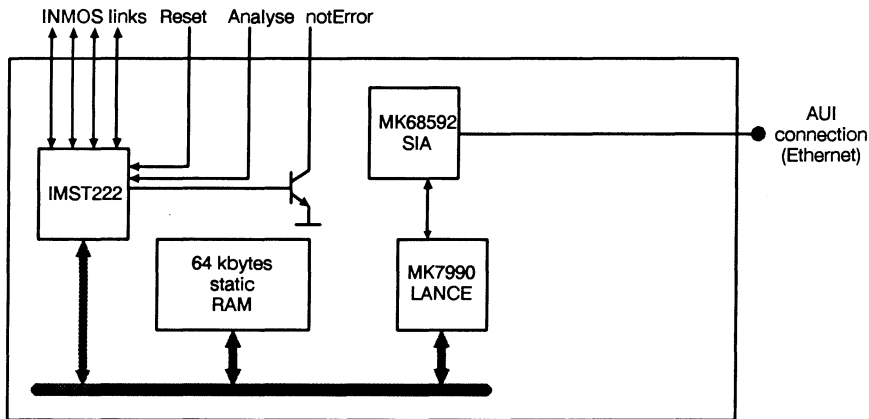
inmos[®]

IMS B431

Ethernet TRAM

Size 2

Product Overview



FEATURES

- Connects transputer systems to IEEE802.3 Local Area Networks (Ethernet)
- IMS T222, 16-bit Transputer
- 64 kbytes SRAM, 100ns access cycle
- Uses MK7990 (LANCE) Ethernet Controller
- Communicates via 4 INMOS serial links
- (Selectable between 10 or 20 Mbits/s)
- Designed to a published specification (*INMOS Technical Note 29*)

GENERAL DESCRIPTION

The IMS B431 is an INMOS link to Ethernet interface. It allows transputer systems to be connected to other computers and computer networks via IEEE802.3 Local Area Networks (LANs). The IMS B431 consists of an IMS T222 16 bit transputer with 64 kbytes of SRAM. The Ethernet interface is implemented with the MK7990 (LANCE) and MK68592. An Attachment Unit Interface (AUI) is provided for connection to Ethernet Media Access Units.

The IMS B431 is software compatible with the earlier IMS B407 TRAM.

36.1 Specification

TRAM feature		Units	Notes
Transputer type	IMS T222-20		
Number of transputers	1		
Number of INMOS serial links	4		
Amount of SRAM	64	kbyte	
Memory "wait states"	0		
Memory cycle time	100	ns	
Subsystem controller	No		
Peripheral circuitry	IEEE802.3 Interface		
Memory Parity	No		
Size (TRAM size)	2		
Length	3.66	inch	
Pitch between pins	3.30	inch	
Width	2.15	inch	
Component height above PCB	13.9	mm	1
Component height below PCB	3.7	mm	2
Weight	50	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	3
Power supply voltage (VCC)	4.75-5.25	Volt	
Power consumption	10	W	4

Table 36.1 IMS B431 specification

Notes:

- 1 This dimension is larger than is normally stated for TRAMs because of the requirement to connect to Ethernet. The quoted height includes the additional height of a 14-way IDC socket, minus strain relief, plugged into the Ethernet AUI connector.
- 2 This dimension includes the thickness of the PCB.
- 3 The figure quoted refers to the ambient air temperature.
- 4 The power consumption is the worst case value obtained when a sample of IMS B431 TRAMs were tested (running a program that utilised all four links and accessed memory simultaneously) at a supply voltage (VCC) of 5.25 V.

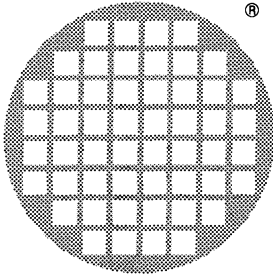
36.2 Ordering Information

Description	Order Number
IMS B431 TRAM with IMS T222-20	IMS B431-10

Table 36.2 Ordering information



Motherboards and other Standard Interface Boards

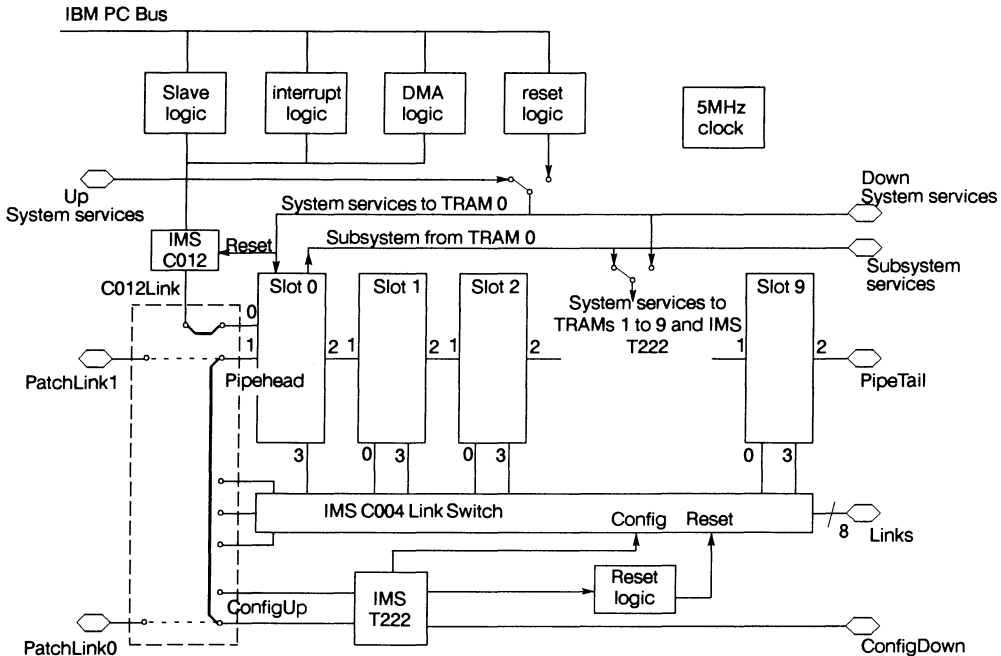


IMS B008

IBM PC Motherboard

inmos[®]

Engineering Data



FEATURES

- IBM PC-AT format board
- Ten transputer module (TRAM) slots
- IBM PC bus interface supports DMA and interrupts
- IMS C004 programmable 32 way crossbar switch allows link configuration
- Conforms to the Module Motherboard Architecture (*INMOS Technical Note 49*)
- 37 way D-type connector gives access to links and system services allowing larger systems to be built

GENERAL DESCRIPTION

The IMS B008 is a TRAM motherboard which plugs into the IBM PC-XT or PC-AT and provides an interface between the IBM PC and transputer based systems. It has slots for up to ten TRAMs. Links 1 and 2 from each of the TRAM slots are hard wired on the IMS B008, such that the TRAMs, when plugged in, form a pipeline of processing elements. The remaining links can be "soft-wired" using an INMOS IMS C004 programmable link switch, incorporated on the IMS B008. This arrangement allows a large variety of networks to be created under software control.

37.1 Description

37.1.1 Introduction

The IMS B008 is a full length PC-AT format card which allows transputer systems to interface to the IBM PC-XT or PC-AT bus. It supports up to ten TRAMS plugged into the slots on the board which are configured into a pipeline. An IMS T222 transputer controlling an IMS C004 link switch enables networks of TRAMs to be configured under software control. A connector on the backpanel of the board gives access to links and system services allowing connections to other IMS B008 boards, or to any board compatible with the link and system service signals. The IBM PC bus interface supports DMA and interrupts. A top view of the IMS B008 is shown in figure 37.1.

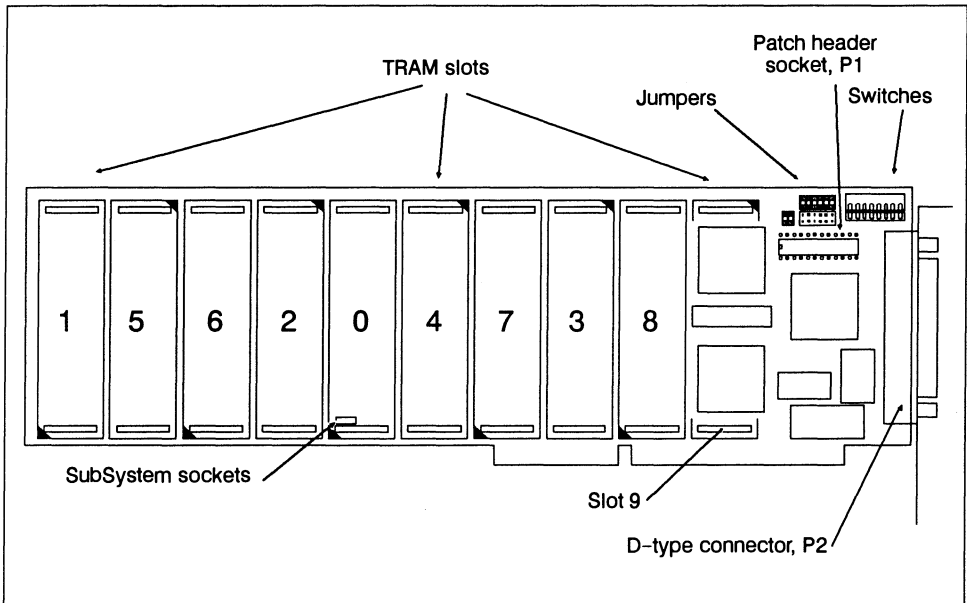


Figure 37.1 Top view of IMS B008

37.1.2 TRAM Slots

The IMS B008 has ten locations for TRAMs to be plugged into, called TRAM slots. Each slot can accommodate a size 1 TRAM. Larger TRAMs may be fitted, occupying more than one slot. Each of the ten slots on the IMS B008 has connections for four INMOS links. Links are numbered 0 to 3 and slots, in the case of the IMS B008, are numbered 0 to 9.

The ten slots on the IMS B008 are connected into a pipeline, using links 1 and 2 from each slot. So slot 0, link 2 is connected to slot 1, link 1; slot 1, link 2 is connected to slot 2, link 1 and so on. The two unconnected links, slot 0, link 1 and slot 9, link 2, at the ends of the slot pipeline are referred to as *pipehead* (slot 0, link 1) and *pipetail* (slot 9, link 2). *Pipetail* is taken out to the 37 way D-type connector, P2, at the back of the board.

Jumpers are provided to allow the IMS B008 to be set up as a head of a pipeline of motherboards or as a board in such a pipeline. Links 0 and 3 from each slot are connected to the IMS C004. If JP1 is fitted then C012Link is connected to slot 0, link 0, and slot 0, link 1 (Pipehead) to IMS T222, link1 (ConfigUp). If JP2 is fitted then Pipehead is connected to PatchLink1, and ConfigUp to PatchLink0. A patch header, inserted into P1, may be used instead of the jumpers, allowing other link wiring options.

In a lot of cases not all of the slots of the IMS B008 will have TRAMs fitted. Even if they are covered by a TRAM they may not be connected to it electrically. In this case to maintain the pipeline connection *pipe-*

jumpers must be used, plugged into each un-occupied slot, or the TRAM covering that slot. These pipe-jumpers connect link 1 to link 2 of the same slot. They are plugged into the pin 1 end of the TRAM slot, with the triangle marked on the corner. The pipejumpers have a mark on them which must be pointing towards the pin 1 marker triangle.

37.1.3 System Services

On all INMOS board products the term 'system services' refers to the collection of the reset, analyse, and error signals. On the IMS B008 the system services for the TRAM in slot 0 can be connected to either the UP system services from another board or the system services controlled by the PC bus interface. System services for the other TRAMs can be connected to the same source as TRAM 0 or to the subsystem port of TRAM 0. As shown in the block diagram the Down and Subsystem services are brought out to the 37 way D-type connector allowing this hierarchy to be extended to multi board systems.

37.1.4 Link Configuration

An IMS T222 transputer and an IMS C004 link switch on the board allow the configuring of the TRAMs into different networks under software control. These networks can also extend onto multiple IMS B008s, or onto other transputer boards, by connecting the links on P2 to the links coming out to an external connector on the other boards.

The configuration information is passed to the IMS T222 either by TRAM 0 when the board is at the head of a pipe of boards or from the ConfigDown link of another board. Link 2 (ConfigDown) from the IMS T222 is taken out to the D-type connector. This allows the IMS T222 devices on all the motherboards in a system to be connected into a pipe allowing configuration information to be passed to each board. The IMS C004 can be hard reset by the IMS T222.

37.1.5 IBM PC Bus Interface

The PC bus has become a de-facto standard after appearing in the IBM PC. Since then a large number of other machines have become available that incorporate the PC bus. The IMS B008 has been designed to work when plugged into either a PC/AT bus slot or a PC bus slot (but the number of options for the interrupt and Direct Memory Access (DMA) channels are reduced when plugged into a PC bus slot).

The bus interface on the IMS B008 performs four functions :

- 1 Converting the 8 bit parallel transfers on the PC bus to serial INMOS link transfers, and vice versa.
- 2 Providing a system services port.
- 3 Controlling DMA transfers.
- 4 Generating interrupts on events on the link interface, when transputer error has been asserted or on DMA transfer end.

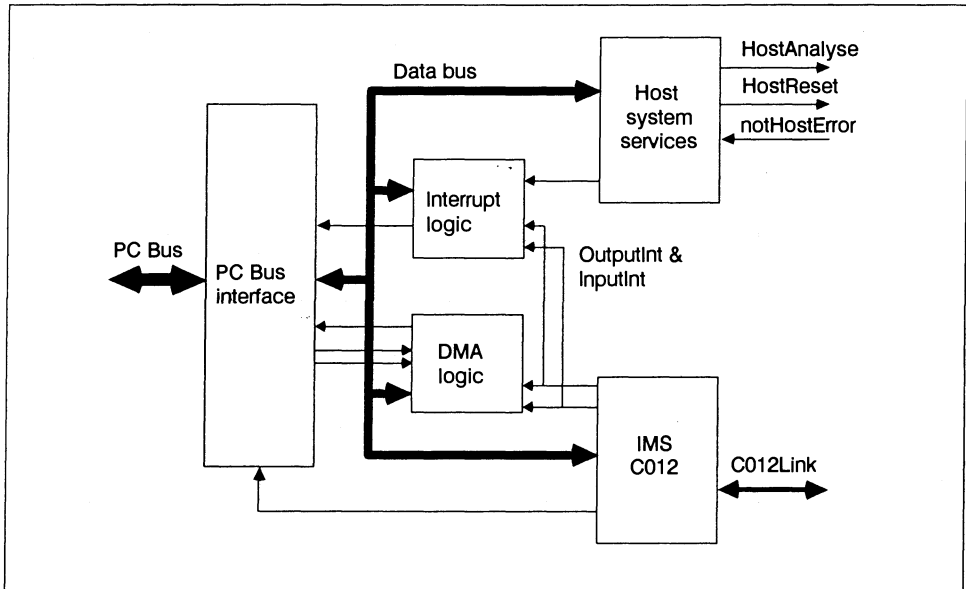


Figure 37.2 PC bus interface block diagram

A block diagram of the PC bus interface is shown in figure 37.2. To enable control of the bus interface functions from software running on the PC, the PC bus interface has a number of registers mapped into the I/O address space of the PC bus (separate from the memory address space).

Link interface

An IMS C012 link adaptor is used as the basis of the link interface on the IMS B008. This device performs the parallel data to serial INMOS link conversions in both directions in a similar fashion to a UART device used on an RS232 interface.

The IMS C012 has four registers which can be written to or read by the PC bus, these registers and the interface control registers occupy a 32 byte area in the address map, the base address of which can be located at 150, 200, or 300 (HEX). A memory map is shown in table 37.1.

Board address	Register
Board base address + #00	IMS C012 Input data register
Board base address + #01	IMS C012 Output data register
Board base address + #02	IMS C012 Input status register
Board base address + #03	IMS C012 Output status register
Board base address + #10	Reset/Error register
Board base address + #11	Analyse register
Board base address + #10	Error location
Board base address + #12	DMA request register
Board base address + #13	Interrupt enable register
Board base address + #14	DMA and interrupt channel select register

Table 37.1 IMS B008 memory map

Host system services

A port is provided by the PC bus interface to allow software on the PC to provide 'system services' to transputers connected to the IMS B008, either as TRAMs plugged into the board or transputers on other boards. The port appears as two registers in the I/O map of the PC.

DMA

DMA logic on the IMS B008 allows data to be transferred between the PC memory and the transputer system at a faster rate than is possible using a polling scheme or interrupting on each byte transferred. DMA requests are generated when the IMS C012 is free to transmit a byte or has received a byte depending on the direction of the transfer. These DMA requests can be generated on DMA channels 0, 1, or 3 with only 1 and 3 being available on the IBM PC-XT. Control of the direction of transfer and starting of the DMA process is achieved by writing into the DMA request register.

Interrupts

The IMS B008 can generate an interrupt on the PC bus when one of the following events occurs:

- The end of a DMA transfer
- notHostError is asserted
- The OutputInt signal from the IMS C012 is asserted
- The InputInt signal from the IMS C012 is asserted

Generation of interrupts is enabled on a particular event when the corresponding enable bit is set in the interrupt enable register. Interrupt channels 3, 5, 11, or 15 can be driven on the PC bus selected by switches on the IMS B008.

37.1.6 Link Speeds

The link speeds of the TRAMs, the IMS C012, and the IMS C004 can be set to 10 or 20 Mbits/s. Link speeds for the IMS T222 link 0 can be set 5, 10, or 20 Mbits/s

37.2 Specifications

Mechanical details

The IMS B008 is a PC/AT format board and is nominally 354mm by 126mm by 22mm overall. The PCB thickness is nominally 1.6 mm. The board includes a metal PC I/O bracket through which the 37 way D-type P2 passes. This bracket serves two functions, to ensure the board is held rigidly at the edge connector end and to maintain the integrity of the shielding of the PC. To enable the bracket to perform these functions it must be securely fixed to the backpanel metalwork of the PC by a screw passing through the slot on the right hand side of the bracket, as viewed with the board towards you in the PC. A mechanical drawing of the IMS B008 is shown in figure 37.3.

The IMS B008 weighs 187g without any TRAMs or a patch header fitted.

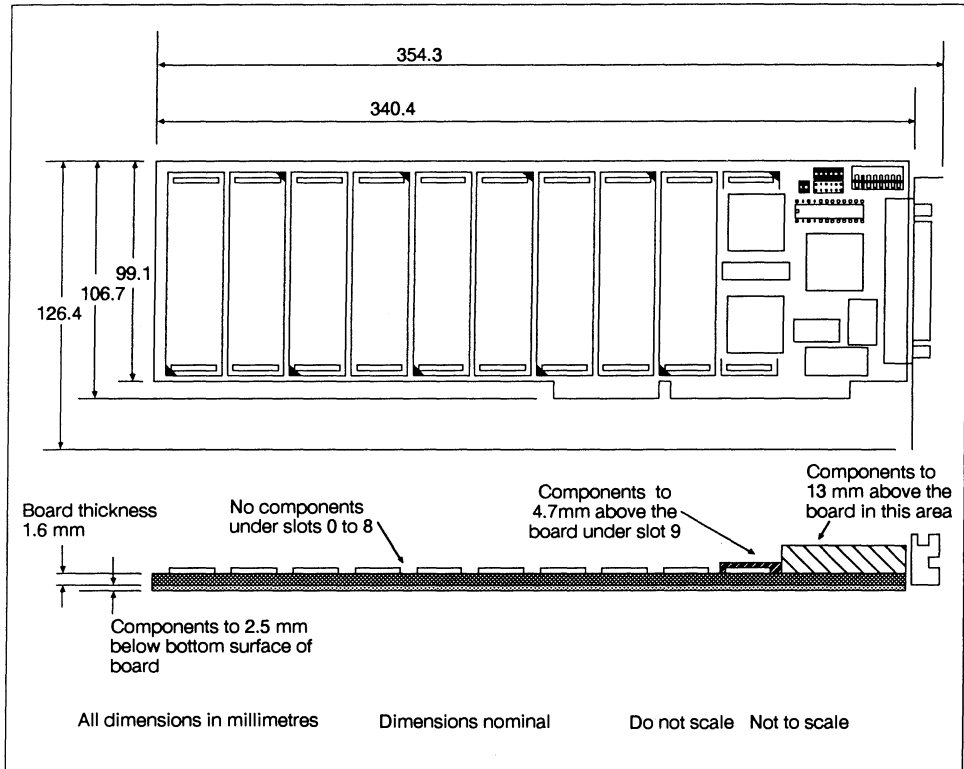


Figure 37.3 Mechanical drawing of the IMS B008

Thermal Information

The IMS B008 with no TRAMs installed will dissipate not more than 3W.

When installing the IMS B008 in a PC consideration needs to be given to cooling airflow not only across the IMS B008 itself but also any TRAMs fitted to it. It is the responsibility of the user to ensure that the operating environment limits for the IMS B008 listed in table 37.2 are not exceeded. This will not occur as long as there are not a large number of high dissipation boards also present in the PC.

To ensure good airflow in the PC the blank backpanels should be present in any slots that are empty.

Operating and Storage Environments

The IMS B008 is designed to be operated and stored in the environments in table 37.2.

Parameter	Operating	Storage
Ambient air temperature	0 to +50°C	-55 to +85°C
Relative Humidity	95% non condensing	95% non condensing
Thermal Shock	<0.08°C/s	<0.15°C/s
Altitude	-300 to +3000m	-300 to +16000m

Table 37.2 Environmental details

Electrical details

The IMS B008 only requires a +5V dc supply which must be between 4.75V and 5.25V with less than 50mV peak-peak noise and ripple between DC and 10MHz. The IMS B008 does not incorporate protection against incorrect power supplies. Major damage will result from connecting a supply to the board which is outside its power supply range. The IMS B008 will draw a maximum of 600mA from the +5V supply.

37.3 Connector Pin Assignments

P1 pin assignments

The pin numbering for this connector is same as for a standard 24 pin IC. The pin assignments are shown in figure 37.4.

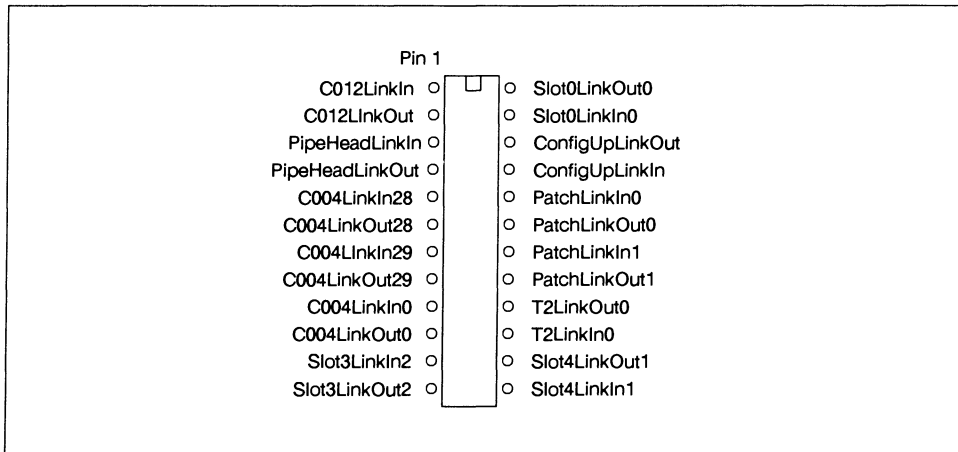


Figure 37.4 P1, the patch header socket pin assignments

P2 pin assignments

notUpReset	20	1	GND
notUpError	21	2	notUpAnalyse
EdgeLinkIn0	22	3	EdgeLinkOut0
EdgeLinkIn1	23	4	EdgeLinkOut1
EdgeLinkOut2	24	5	GND
EdgeLinkOut3	25	6	EdgeLinkIn2
EdgeLinkOut4	26	7	EdgeLinkIn3
GND	27	8	EdgeLinkIn4
EdgeLinkIn5	28	9	EdgeLinkOut5
EdgeLinkIn6	29	10	EdgeLinkOut6
EdgeLinkIn7	30	11	EdgeLinkOut7
PatchLinkOut0	31	12	GND
PatchLinkOut1	32	13	PatchLinkIn0
notSubSystemReset	33	14	PatchLinkIn1
notSubSystemError	34	15	notSubSystemAnalyse
PipeTailLinkIn	35	16	PipeTailLinkOut
ConfigDownLinkIn	36	17	ConfigDownLinkOut
notDownAnalyse	37	18	notDownReset
		19	notDownError

Figure 37.5 Pin assignments for the 37 way D-type connector, P2

37.4 Jumpers

There are two 12 pin (JP1 and JP2) and one 4 pin (JP3) jumper pin arrays on the IMS B008. Jumper sockets which connect two pins together should be installed in all positions on a pin array if a jumper is to be used. This requires six jumper sockets to be used on jumper pin arrays JP1 and JP2, and two on JP3. Fitting of jumper sockets onto the pin arrays is shown in figure 37.6. Jumpers must be fitted with their long side parallel to the backpanel of the IMS B008.

Note that the jumpers should be fitted with the small square holes in the plastic down towards the board. Use of jumpers JP1 and JP2 are mutually exclusive and the use of any of the jumpers is mutually exclusive with the use of a patch header plugged into P1. The connections made by the jumpers are shown in table 37.3.

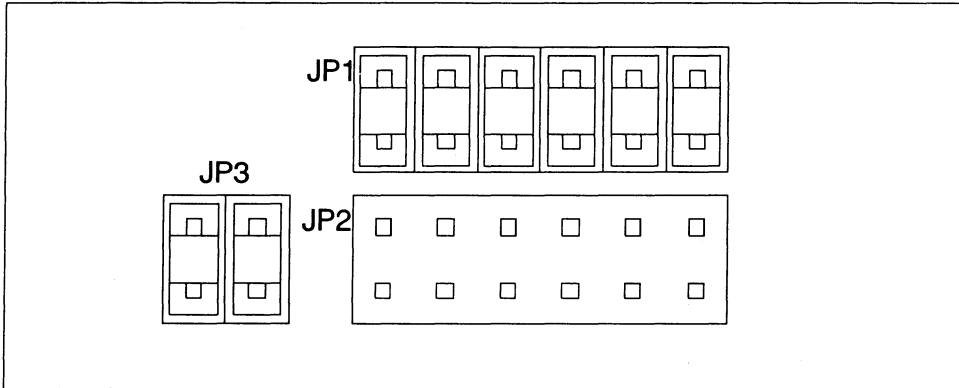


Figure 37.6 Diagram showing the fitting of jumpers to the jumper pin arrays, jumpers shown fitted to JP1 and JP3

Jumper pin array	Connections made when jumper sockets fitted
JP1	C012Link to slot 0, link 0 slot 0, link 1 to IMS T222, link 1 IMS T222, link 0 to IMS C004, link 28
JP2	Pipehead to PatchLink1 ConfigUp to PatchLink0 slot 0, link 0 to IMS C004, link 28
JP3	slot 3, link 2 to slot 4, link 1

Table 37.3 Connections made by the jumper pin arrays

37.5 Switches

There are eight switches on the IMS B008 which select options for the PC interface, system services, and the link speeds. The functions of each of these switches is shown in tables 37.4 to 37.9.

Note a switch is on when the slider is pushed towards the top of the board, away from the edge connector, as shown in figure 37.7.

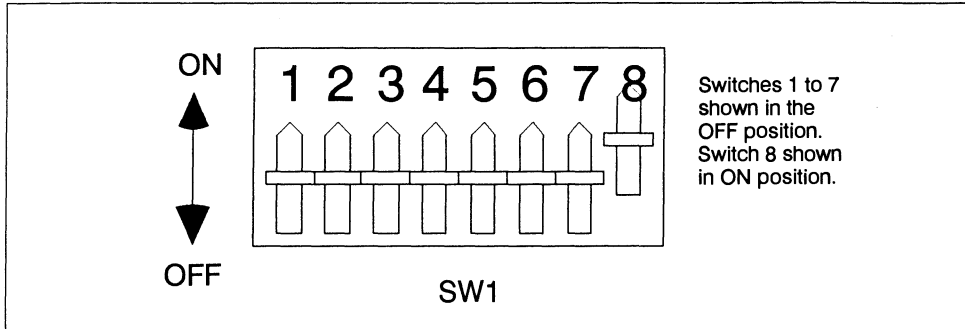


Figure 37.7 Detail of switch on the IMS B008 showing the ON and OFF positions.

Switch No.	Function
SW1:1	PC bus interface enable and board base address selection
SW1:2	PC bus interface enable and board base address selection
SW1:3	Slot 0 and Down system services selection
SW1:4	Slots 1 to 9 system services selections
SW1:5	Link speed selections
SW1:6	Link speed selections
SW1:7	Link speed selections
SW1:8	Link speed selections

Table 37.4 Switch functions when ON and OFF

SW1:1	SW1:2	Board base address
ON	ON	PC bus interface disabled
ON	OFF	#150
OFF	ON	#200
OFF	OFF	#300

Table 37.5 Board base address and PC bus interface disable selections

SW1:3	TRAM slot 0 and DOWN system services
ON	From Up system services
OFF	From Host system services

Table 37.6 Selections for slot 0 and Down system services

SW1:4	TRAM slots 1 - 9 system services
ON	From TRAM in slot 0 Subsystem services
OFF	From Down system services

Table 37.7 System services selections for slots 1 to 9

SW1:5	SW1:6	Link 0 speed	Links 1 - 3 speed	Notes
ON	ON	10 Mbits/s	10 Mbits/s	
OFF	ON	10 Mbits/s	20 Mbits/s	1
ON	OFF	20 Mbits/s	10 Mbits/s	2
OFF	OFF	20 Mbits/s	20 Mbits/s	

Table 37.8 TRAM link speed selections

- Note that this switch setting results in an inoperable link 3 since this link is connected to the IMS C004 which will be set to 10 Mbits/s by this setting.
- This switch setting results in an inoperable link 3 since the IMS C004 will be set to a link speed of 20 Mbits/s.

SW1:6	SW1:7	SW1:8	C012 and C004 Link(s)	T2 Link 0	T2 Links 1 - 3	Notes
ON	ON	ON	10 Mbits/s	10 Mbits/s	10 Mbits/s	
ON	ON	OFF	10 Mbits/s	5 Mbits/s	10 Mbits/s	
ON	OFF	ON	10 Mbits/s	10 Mbits/s	10 Mbits/s	
ON	OFF	OFF	10 Mbits/s	20 Mbits/s	10 Mbits/s	
OFF	ON	ON	20 Mbits/s	10 Mbits/s	5 Mbits/s	1
OFF	ON	OFF	20 Mbits/s	5 Mbits/s	5 Mbits/s	1
OFF	OFF	ON	20 Mbits/s	10 Mbits/s	20 Mbits/s	
OFF	OFF	OFF	20 Mbits/s	20 Mbits/s	20 Mbits/s	

Table 37.9 IMS C012, IMS C004, and IMS T222 link speed selections

- Note these switch settings will result in the T2 not being able to send configuration messages to the IMS C004 link switch.

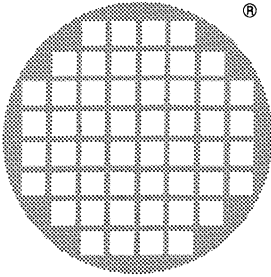
37.6 Reference

- IMS B008 User Guide and Reference Manual, Inmos Limited, 1990

37.7 Ordering Information

Description	Order Number
IMS B008 IBM PC Module Motherboard	IMS B008-1

Figure 37.8 Ordering information

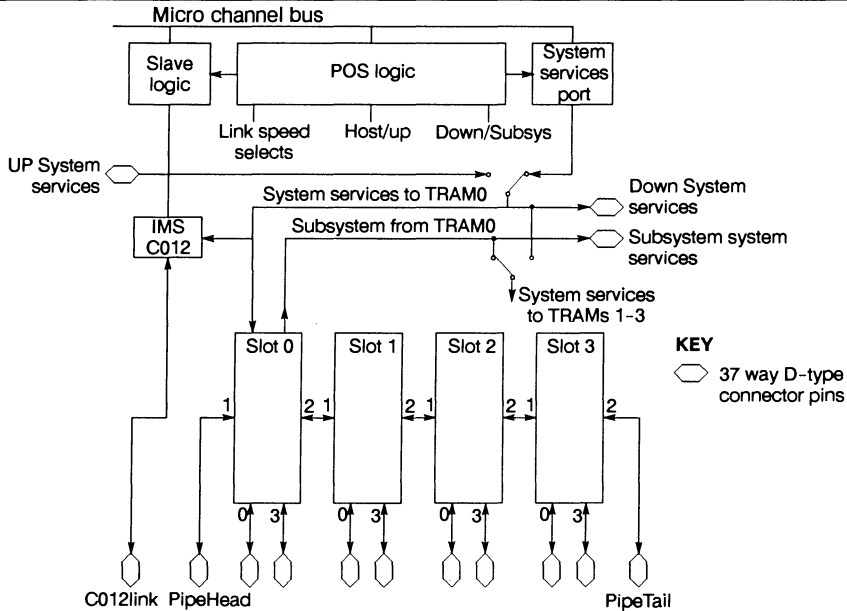


inmos[®]

IMS B017

IBM PS/2 Motherboard

Engineering Data



FEATURES

- IBM PS/2 Micro Channel bus format board
- Four TRAM slots accommodating size 1 or size 2 transputer modules
- Provides a gateway to larger transputer networks from MCA bus based systems
- Link adapter interface to the MCA bus

GENERAL DESCRIPTION

The IMS B017 is a **TR**Aansputer **M**odule (TRAM) motherboard designed to plug into a Micro Channel bus. The board has four TRAM slots and an interface to the Micro Channel bus.

The Micro Channel bus interface provides a single INMOS serial link and a system services

port. Software running on the Micro Channel based system can reset, analyse, communicate with, and monitor the error flag of a transputer network on or connected to the IMS B017. Data can be transferred to and from the link interface using programmed I/O.

The IMS B017 TRAM slots are connected into a pipeline using two of the four links from each slot. The remaining two links from each slot, and the pipeline head and tail links are connected to a 37 way D-type connector. This allows the links from each slot to be connected to each other, or to the links from other motherboards, to form transputer networks other than a pipeline.

SGS-THOMSON
MICROELECTRONICS

INMOS is a member of the SGS-THOMSON Microelectronics group

38.1 IMS B017 engineering data

38.1.1 Description

TRAM motherboards provide a number of slots into which TRAMs can be plugged. Each of these slots provides the necessary connections to power, clock, reset signals and the INMOS links. The motherboard provides a method of connecting TRAMs together and may also include special circuitry to provide an interface to something other than a transputer system. In the case of the IMS B017 this is an interface to the Micro Channel bus. These motherboards can be used to build networks of transputers of arbitrary size and are supported by a range of software products from INMOS.

The IMS B017 has four TRAM slots and is able to accommodate both size 1 and size 2 TRAMs. The interface to the Micro Channel bus includes a single INMOS link and a system services port. A link adapter is used to convert from the serial INMOS link format to parallel Micro Channel bus format and vice versa. Software running on the Micro Channel system can communicate with, analyse, reset and monitor the error flag of a transputer network on or connected to the IMS B017. Data can be transferred to and from the link interface using programmed I/O. Interrupts can be generated on link events or on error being asserted, freeing the processor from polling the IMS B017 to detect these events.

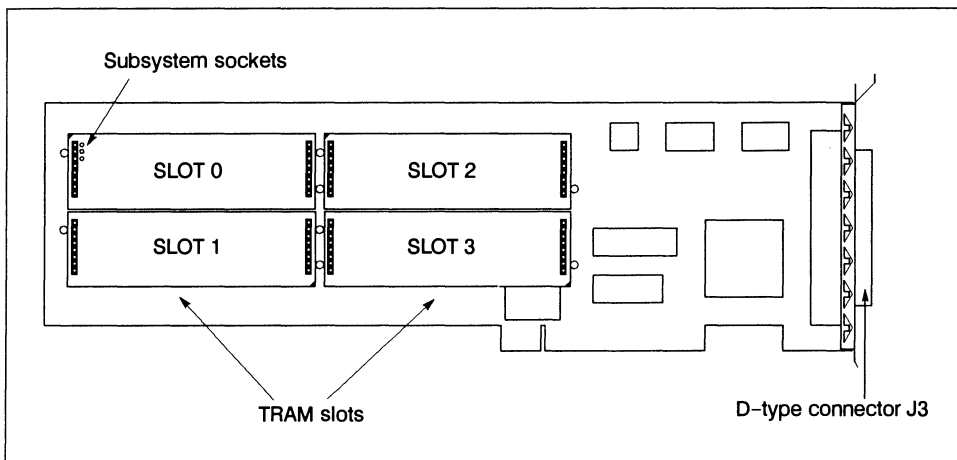


Figure 38.1 Top view of the IMS B017

38.1.2 TRAM Slots

The IMS B017 has four locations for TRAMs to be plugged into, called TRAM slots. Each slot can accommodate a size 1 TRAM. Size 2 TRAMs can be fitted, occupying two slots. Each of the four slots on the IMS B017 has connections for four INMOS links. Links are numbered 0 to 3 and slots, in the case of the IMS B017, are numbered 0 to 3.

The four slots on the IMS B017 are connected into a pipeline, using links 1 and 2 from each slot. So slot 0, link 2 is connected to slot 1, link 1; slot 1, link 2 is connected to slot 2, link 1 and so on.

In some cases not all of the slots of the IMS B017 will have TRAMs fitted. Even if they are covered by a TRAM they may not be connected to it electrically. In this case to maintain the pipeline connection *pipejumpers* must be used, plugged into each un-occupied slot, or the TRAM covering that slot. These pipejumpers connect link 1 to link 2 of the same slot. They are plugged into the pin 1 end of the TRAM slot, with the triangle marked on the corner. The pipejumpers have a mark on them which must be pointing towards the pin 1 marker triangle.

The two unconnected links, slot 0, link 1 and slot 3, link 2, at the ends of the slot pipeline are referred to as *pipehead* (slot 0, link 1) and *pipetail* (slot 3, link 2). *Pipetail*, *pipehead*, links 0 and 3 from each slot, and the link from the IMS C012 are taken out to the 37 way D-type connector, J3, at the back of the board.

By connecting links together on J3, networks of transputers can be set up on the IMS B017. These networks can also extend onto multiple IMS B017s, or onto other transputer boards, by connecting the links on J3 to the links coming out to an external connector on the other boards.

The INMOS link connections between the slots, J3, and the C012Link from the IMS C012 are shown in figure 38.2.

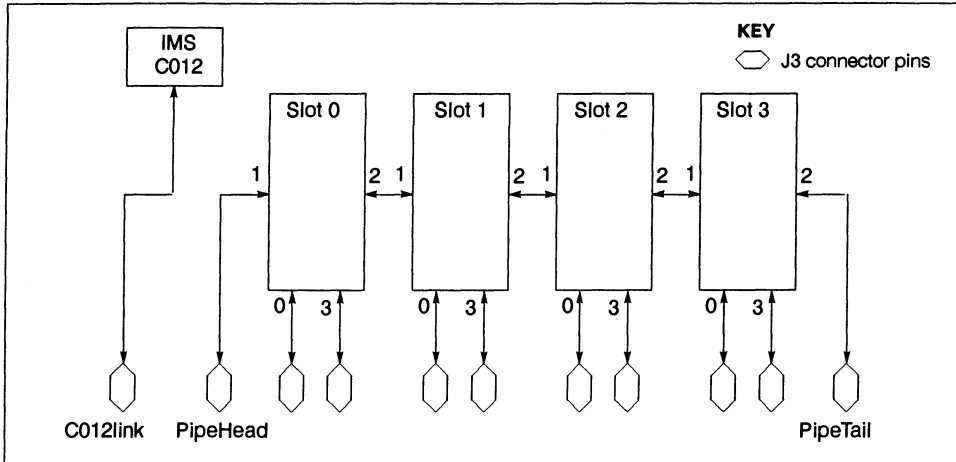


Figure 38.2 INMOS link connections on the IMS B017

38.1.3 System Services

On all INMOS board products the term 'system services' refers to the collection of the reset, analyse and error signals. On the IMS B017 the system services for the TRAM in slot 0 can be connected to either the UP system services from another board or the system services controlled by the PS/2 bus interface. System services for other TRAMs can be connected to the same source as TRAM 0 or to the subsystem port of TRAM 0. As shown on the block diagram the Down and Subsystem services are brought out to the 37 way D-type connector, allowing this hierarchy to be extended to multi board systems.

38.1.4 Micro Channel bus interface

The Micro Channel (MC) bus has become a de-facto standard after appearing in the IBM Personal System/2 (PS/2) machines. Since then a number of other machines have been released that incorporate the Micro Channel bus. The IMS B017 has been designed to work when plugged into a Micro Channel bus slot in a PS/2 but should also operate correctly in other Micro Channel based systems.

The bus interface on the IMS B017 performs four functions :

- 1 Providing the Programmable Option Select (POS) registers used by the PS/2 setup utilities in controlling the configuration of the IMS B017.
- 2 Converting the 8 bit parallel transfers on the Micro Channel bus to serial INMOS link transfers, and vice versa.
- 3 Providing a system services port.
- 4 Generating interrupts on events on the link interface or when transputer error has been asserted.

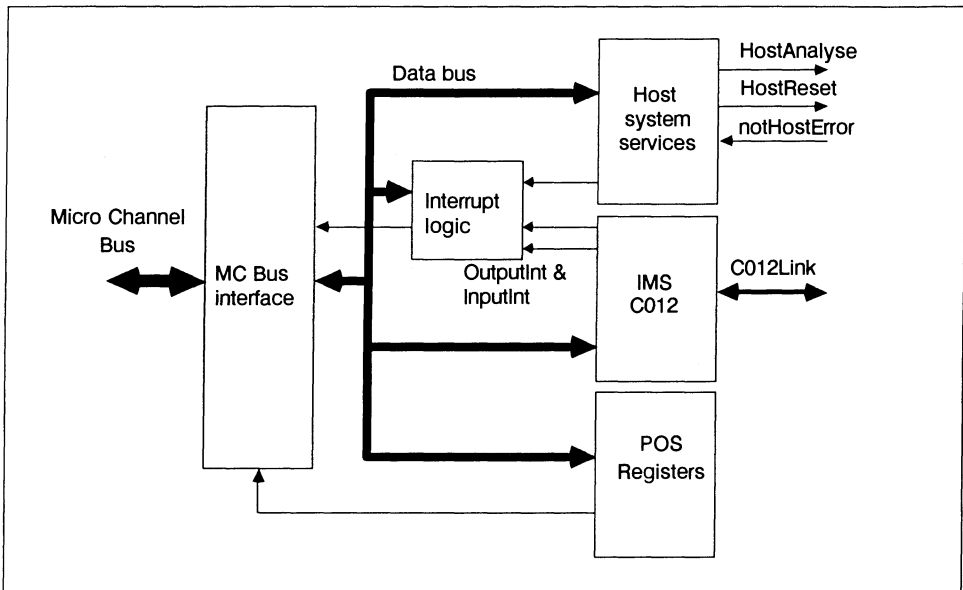


Figure 38.3 Micro Channel bus interface block diagram

A block diagram of the Micro Channel bus interface is given in figure 38.3. To enable control of the bus interface functions from software running on the Micro Channel based system, the MC bus interface has a number of registers mapped into the I/O address space of the Micro Channel bus (separate from the memory address space). Details of these are given in [1].

Link interface

An IMS C012 link adaptor is used as the basis of the link interface on the IMS B017. Detailed information on this device can be found in [2]. This device performs the parallel data to serial INMOS link conversions in both directions in a similar fashion to a UART device used on an RS232 interface. The link coming from the link adaptor is labelled C012Link in figure 38.3. The IMS C012 has four registers which can be written to or read by the Micro Channel bus, more information is given in [1].

Host system services

A port is provided by the Micro Channel bus interface to allow software on the MC based system to provide 'system services' to transputers connected to the IMS B017, either as TRAMs plugged into the board or transputers on other boards. The port appears as two registers in the I/O map of the Micro Channel. In addition an extra register, the *Error Status/Interrupt Control register* allows the state of the notError signals from each of the TRAM slots to be monitored. The system services port and the error status register function is described in [1].

Interrupts

The IMS B017 can generate an interrupt on the Micro Channel bus when one of the following events occurs:

- notHostError is asserted
- The OutputInt signal from the IMS C012 is asserted
- The InputInt signal from the IMS C012 is asserted

Generation of interrupts on each of these events is controlled by two interrupt enable bits in the Output Status register and Input Status register, and bit 0 of the *Error Status/Interrupt Control register*. Setting a bit to one in one of these registers enables interrupts on the event corresponding to that bit.

38.1.5 Configuration

Configuration of the IMS B017 is carried out by installing any TRAMs required on the board, selecting options by using the system configuration utilities supplied by IBM with the PS/2, and by making connections to the 37 way D-type, J3, on the back of the board. Further information on setting the IMS B017 configuration is given in [1].

38.1.6 Specifications

Mechanical Details

The IMS B017 is a PS/2 Micro Channel adaptor format board and is nominally 322mm by 108mm by 21mm overall. The PCB thickness is nominally 1.6 mm. The board includes a metal PS/2 I/O bracket through which the 37 way D-type J3 passes. This bracket serves two functions, to ensure the board is held rigidly at the edge connector end and to maintain the integrity of the shielding of the PS/2. To enable the bracket to perform these functions it must be securely fixed to the backpanel metalwork of the PS/2 by a screw on the outside of the case.

The IMS B017 weighs 150g without any TRAMs fitted.

Thermal Information

The IMS B017 with no TRAMs installed will dissipate not more than 2W.

When installing the IMS B017 in a PS/2 consideration needs to be given to cooling airflow not only across the IMS B017 itself but also any TRAMs fitted to it. It is the responsibility of the user to ensure that the operating environment limits for the IMS B017 listed in table 38.1 are not exceeded. This will not occur as long as there are not a large number of high dissipation boards also present in the PS/2.

To ensure good airflow in the PS/2 the blank backpanels should be present in any slots that are empty.

Operating and Storage Environments

The IMS B017 is designed to be operated and stored in the environments in table 38.1.

Parameter	Operating	Storage
Ambient air temperature	0 to +50°C	-55 to +85°C
Relative Humidity	95% non condensing	95% non condensing
Thermal Shock	<0.08°C/s	<0.15°C/s
Altitude	-300 to +3000m	-300 to +16000m

Table 38.1 Environmental details

Electrical Details

The IMS B017 only requires a +5 V dc supply which must be between 4.75V and 5.25V with less than 50mV peak-peak noise and ripple between DC and 10MHz. The IMS B017 does not incorporate protection against incorrect power supplies. Major damage will result from connecting a supply to the board which is outside its power supply range. The IMS B017 with no TRAMs installed will draw a current of no more than 400mA.

The maximum power consumption of the IMS B017 is 2W. The power loading specification of the Micro Channel bus should be adhered to when considering which TRAMs are to be fitted.

38.1.7 Memory Map

Addresses	Register
Board base address + #00	Link adaptor input data register

Board base address + #01	Link adaptor output data register
Board base address + #02	Link adaptor input status register
Board base address + #03	Link adaptor output status register
Board base address + #10	Reset/Error register
Board base address + #11	Analyse register
Board base address + #12	Error status/Interrupt control register

- 1 Board base address can be set to #0150 or #0200

Table 38.2 IMS B017 memory map

38.1.8 Connector Pin Assignments

J3 pin assignments

		1	GND
notUpReset	20	2	notUpAnalyse
notUpError	21	3	Slot0Out0
Slot0In0	22	4	Slot0Out3
Slot0In3	23	5	GND
Slot1Out0	24	6	Slot1In0
Slot1Out3	25	7	Slot1In3
Slot2Out0	26	8	Slot2In0
GND	27	9	Slot2Out3
Slot2In3	28	10	Slot3Out0
Slot3In0	29	11	Slot3Out3
Slot3In3	30	12	GND
C012LinkOut	31	13	C012LinkIn
PipeHeadOut	32	14	PipeHeadIn
notSubSystemReset	33	15	notSubSystemAnalyse
notSubSystemError	34	16	PipeTailOut
PipeTailIn	35	17	NC
NC	36	18	notDownReset
notDownAnalyse	37	19	notDownError

Figure 38.4 Pin assignments for the 37 way D-type connector, J3

38.1.9 References

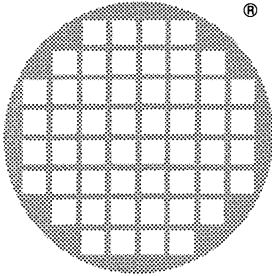
- 1 IMS B017 User Guide and Reference Manual, Inmos Limited, 1990
- 2 The Transputer Databook, second edition, IMS C012 engineering data, Inmos Limited, 1990
- 3 *Personal System/2 Hardware Interface Technical Reference*, International Business Machines Corporation, 1988.

38.1.10 Ordering Information

Description	Order number
IMS B017 IBM PS/2 Motherboard	IMS B017-1*
IMS S217:IMSB017 driver for PS/2	Included in IMSB017

Table 38.3 Ordering Information

*Includes IMS S217 driver for PS/2

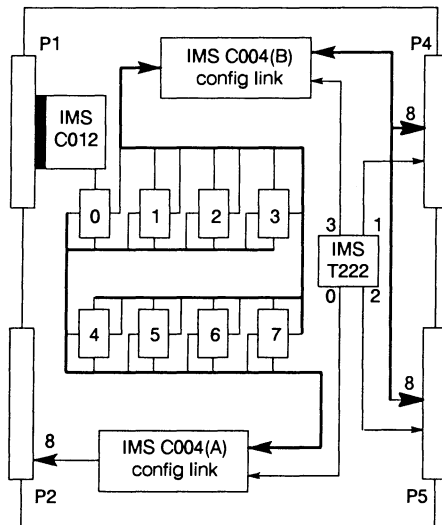


inmos[®]

IMS B014

VMEbus slave card

Engineering Data



FEATURES

- Compatible with VMEbus Specification Rev. C.1
- Accommodates 8 standard transputer modules (TRAMs)
- Static or dynamic link configuration using two IMS C004 link switches
- VMEbus interface designed around an IMS C012 link adaptor
- Expandable to form arbitrarily large systems
- Suitable for use as VMEbus-transputer interface with a SUN based development system and IMS CA12 card frame

GENERAL DESCRIPTION

The IMS B014 module motherboard is compatible with VMEbus Specification Rev. C.1. It is a standard depth (160mm), double height (6U) card, containing 8 TRAM slots with associated configuration circuitry and a VMEbus slave interface. Two IMS C004 crossbar link switches are provided to allow the user to configure the transputer link connections. This architecture allows any topology to be established on the board. Additionally, 24 links are brought to the edge connectors (8 on the P2 back connector, and 16 split between two front connectors) so that larger networks, using multiple boards, may be constructed.

39.1 Description

39.1.1 VMEbus Interface

The IMS B014 has a slave interrupting interface to the VMEbus. This interface provides access to a single, bi-directional INMOS link and a system service port. The interface appears as a number of registers located in the A16 (short) address space on the VMEbus, which may be accessed by any VMEbus master such as the IMS B016. These registers are used to program and interact with the IMS B014.

The TRAMS on the IMS B014 can be reset or analysed via the VMEbus interface, or can be bootstrapped through it. Data can be exchanged between TRAMS on the IMS B014 and any bus-master on the VMEbus. All bus communication is achieved using D08(O) data transfers.

39.1.2 Interrupts

The IMS B014 is capable of generating a single VMEbus interrupt that may be assigned to any of the seven VMEbus priority interrupt levels. Interrupts can be triggered by any one of three events:

- data byte received on VMEbus link;
- VMEbus link free to send a data byte;
- an error has occurred in the transputer system;

All interrupts may be individually masked.

39.1.3 IMS C004 Control

The IMS B014 uses the same method of controlling the IMS C004 as other INMOS module motherboards. This allows all IMS C004s to be programmed from a single master configuration link. Each module motherboard has a "config-up" link and a "config-down" link. Thus, motherboards may be cascaded to build multi-board systems, by connecting these links in a pipeline.

On the IMS B014, the "config-up" and "config-down" links can be switched to either the P2 back connector or to the front connectors (P4, P5). Jumpers are also provided that allow either the VMEbus link or slot 0, link 1 to be the master configuration link (figure39.1).

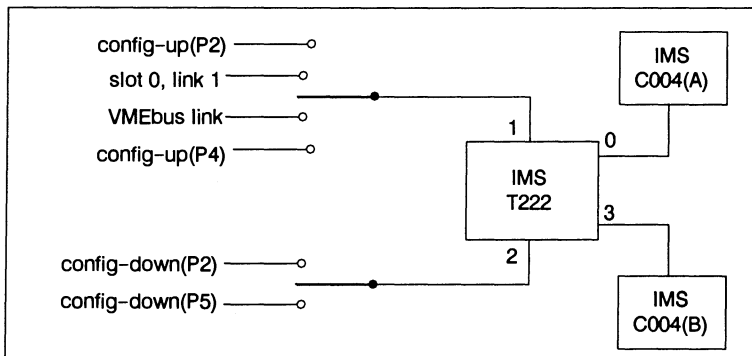


Figure 39.1 Configuration Control

39.1.4 System Services Organisation

On all INMOS board products, the term "system services" refers to the collection of the reset, analyse and error signals. "Reset-up" and "Reset-down" ports are used to carry these signals between boards. Error signals "flow" in the reverse direction to the reset and analyse signals.

The IMS B014 allows system service signals to be generated by bus-masters on the VMEbus. A bus-master can reset or analyse the transputer system by writing to the appropriate registers in the interface. Transputer error signals are propagated back to a register in the interface where they may be monitored by the bus-master.

TRAM slot 0 can be reset independently of the other TRAM slots on the board. This allows slot 0 to be used as a "transputer master", controlling other transputers in the system. Thus, it is possible to establish a control hierarchy; a principle that can be extended to multi-level systems using multiple motherboards.

39.2 Specification

Mechanical details

The IMS B014 is designed to accord with DIN 41494 and IEC 297 standards. The board is nominally 160mm by 233.35mm. Nominal board thickness is 1.6mm. The supplied front panel width is 4HP (approx 20mm). This is compatible with a board-to-board pitch in a card cage of 0.8". M2.5 fastening bolts are provided on the front panel, these mate with tapped holes in the card cage and fix the board securely. Front panel handles allow the board to be removed from the card cage (by un-screwing the retaining bolts and pulling hard on the handles). Note that the front panel is *required* when operating the IMS B014 in a card cage, both for mechanical rigidity and to give correct cooling air flow.

Thermal details

Adequate cooling air flow must be provided to maintain the components on the board within their operating temperature. Air flow should run parallel to the board surface and parallel to the front panel. The amount of heat dissipated by the board depends upon the TRAMs fitted. With no TRAMs the IMS B014 dissipates no more than 5W. With TRAMs fitted, the maximum dissipation allowed (from 5v supply) is 18.75W when only using a J1 backplane and 37.5W when using a J1/J2 backplane¹. It is essential that the user ensures that the maximum power dissipation is not exceeded. The cooling air flow required for a particular application will probably need to be determined empirically.

A single board operating in static air at room temperature (and not in a card-cage) will usually not need forced air cooling. This kind of set-up should only be used for lab and development work. High reliability is not to be expected from boards which are not provided with adequate cooling.

Operating and Storage Environments

	Operating	Storage
Temperature	0 to +50°C ambient air	-55 to +85°C
Relative Humidity	95% non condensing	95% non condensing
Thermal Shock	<0.08°C/s	<0.15°C/s
Altitude	-300 to +3000 m	-300 to +16000 m

Table 39.1 Environmental details

Electrical details

The IMS B014 requires power supply voltages in accordance with the VMEbus specification. That is, the +5V dc supply must be between 4.75V and 5.25V and have less than 50mV pk-pk noise and ripple between dc and 10MHz. The IMS B014 does not incorporate protection against incorrect power supplies. Major damage can result from operating the board outside its power supply range.

The maximum power consumption of the IMS B014 without any TRAMs fitted is 10W.

1. J1 is the minimum VMEbus backplane and mates with P1 connectors on VMEbus boards. J2 mates with P2 connectors and is sometimes called a 32-bit backplane because it is needed for 32-bit VMEbus operations. Combined J1/J2 backplanes mate with both P1 and P2 and are needed for reliable operation of fast 32-bit VMEbus transfers.

VMEbus capability

For easy description of VMEbus boards, the VMEbus specification defines a number of “capability” abbreviations. The relevant capabilities for the IMS B014 are listed here—

1. A16:D08(O) SLAVE
2. INT(1-7):D08(O) INTERRUPTER
3. 6U high—double height board

VMEbus access time will be no longer than 170ns from DSA* to DTACK*.

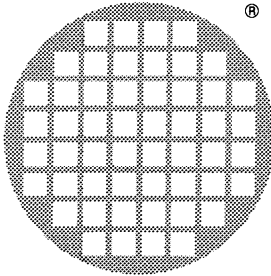
The time to propagate a non-participating interrupt acknowledge cycle is no longer than 100ns from IACK-IN* to IACKOUT*. The time to respond to a participating interrupt acknowledge cycle will be no longer than 170ns from IACKIN* to DTACK*.

The IMS B014 propagates the BUSREQ* daisy-chain signals on the board, thus there is no need for jumpers on the backplane at the slot which the IMS B014 is plugged into.

39.3 Ordering Information

Description	Order Number
VMEbus Module Motherboard with IMS T222	IMS B014-1
6U to 9U VME card frame adapter for SUN	IMS CA12
Device driver software	IMS S514

Table 39.2 Ordering Information

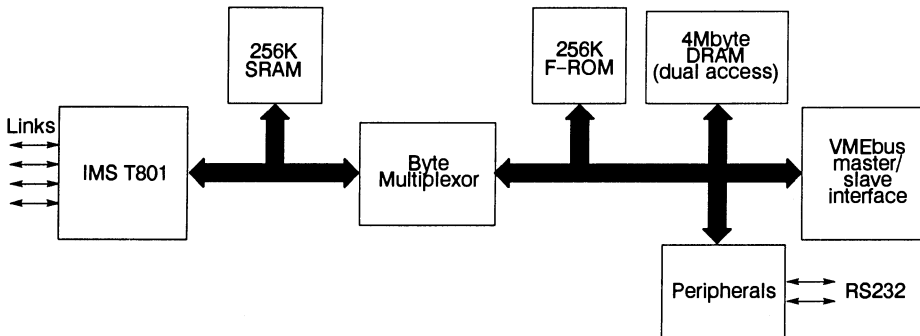


inmos[®]

IMS B016

VMEbus Master/Slave Board

Engineering Data



FEATURES

- VMEbus VIC interface chip
- IMS T801-25 or IMS T801-20 transputer
- 4 INMOS 20Mbits/sec links for direct connection to transputer networks
- Byte multiplexor between T801 and VMEbus allows fast byte reordering to overcome endian incompatibilities
- 256Kbytes private transputer SRAM (8ns cycle)
- 4Mbytes DRAM dual-ported between IMS T801 and VMEbus
- 256 Kbytes of Flash ROM programmable with boot code for transputer
- Full VMEbus interrupter and interrupt handler
- Real Time clock for time of day. When power supply fail, the RTC is automatically switched to the VMEbus +5V standby rail
- 2 RS232 serial ports using a 2681 DUART

GENERAL DESCRIPTION

The IMS B016 is a very high performance VMEbus master/slave board suited to all applications requiring fast data throughput between a transputer network and VMEbus peripherals. The board incorporates a 32-bit transputer processor, local RAM, peripherals and interface circuitry for efficient communication between the transputer and other VMEbus boards. It is designed to give as flexible and efficient interface between the VMEbus and transputers as current technology allows.

Sustained transfer rates in excess of 15Mbytes/s are achievable, given a fast VMEbus system. The IMS T801 on-board is capable of 12.5 MIPS (sustained) and has its own private fast SRAM for speed-critical code and data.

Dual access RAM memory is provided for access by both the transputer and other VMEbus Masters. The transputer, an IMS T801, can perform Master accesses to other VMEbus slaves. It can also interrupt other VMEbus Interrupt Handlers and itself handle VMEbus interrupts.

40.1 Introduction

The IMS B016 is a VMEbus board incorporating a 32-bit transputer processor, local RAM, peripherals, and interface circuitry to allow efficient communication between the transputer and other VMEbus boards. Applications include using the board to create a VMEbus subsystem or as an interface from a host computer.

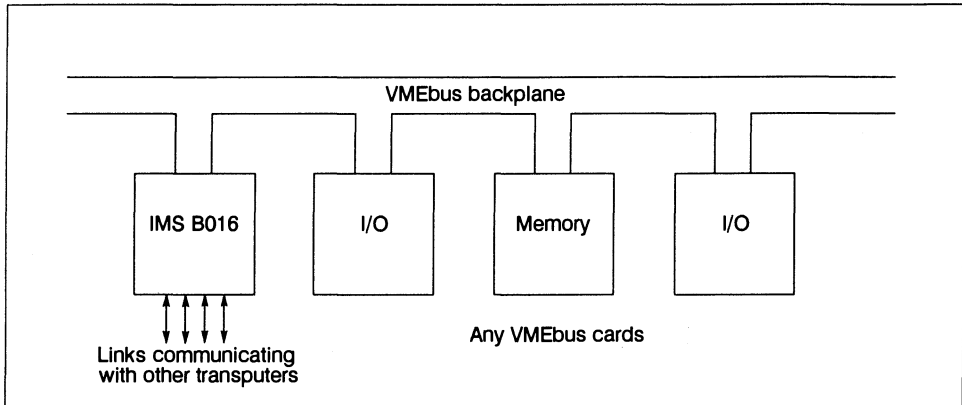


Figure 40.1 The IMS B016 used to create a VMEbus I/O subsystem

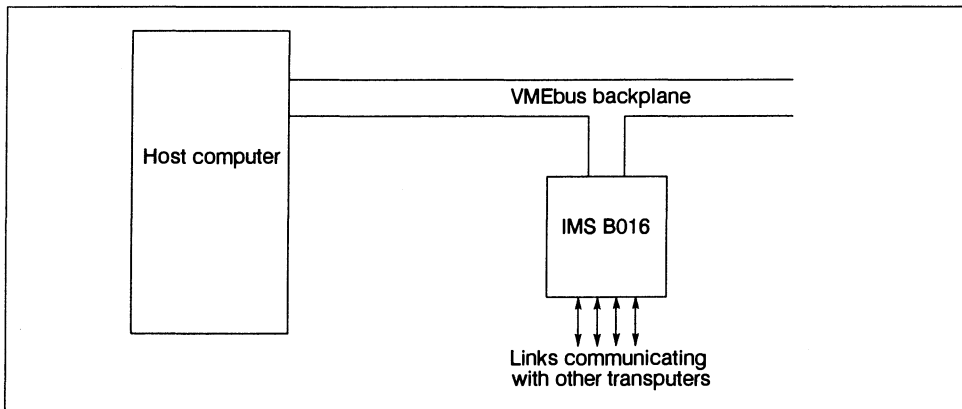


Figure 40.2 The IMS B016 used as an interface from a host computer

VME Bus

The VMEbus, originally proposed by Motorola, Mostek and Philips, is now an IEC and IEEE standard. It provides a parallel 8, 16 or 32-bit bus with multi-master capabilities. Mechanical constraints are basically those of IEC 297 (eurocard).

The IMS B016, through its use of the VIC *VMEbus Interface Chip*, provides the capability to use almost all of the features of the VMEbus. In particular, in slave mode, the card supports BLT (Block Transfer) cycles, RMW (read-modify-write) cycles and UAT (unaligned) cycles. It also functions correctly in a system containing location monitors and performing address pipelining. Write posting is supported for both slave and master accesses.

IMS T801 Transputer

The IMS T801¹ transputer is a 32-bit CMOS microcomputer with a 64-bit floating point unit and graphics support. It has 4Kbytes of on-chip RAM for high speed processing, a 32-bit non-multiplexed internal memory interface and four standard INMOS links.

INMOS links are special serial communications links which allow transputers to talk to each other. Links use two wires to send bidirectional data between two transputers (or other chips) at up to 20Mbits/s. All communication between the IMS T801 transputer on the IMS B016 and other transputers on other cards is via links.

40.2 Description

40.2.1 IMS T801 and private SRAM

The on-board transputer, an IMS T801 floating point processor, has four INMOS serial links. These links allow connections to be made to other transputer devices on other boards via the P2 connector. 256 Kbytes of fast memory (the *private SRAM*) is directly connected to the IMS T801. This cycles in 80ns and is only accessible by the transputer. This memory is typically used to store transputer code and data which does not need to be accessed from the VMEbus. The SRAM is about twice as fast as the next fastest available memory, as seen from the transputer, so its use is recommended whenever possible. The SRAM occupies the first 256Kbytes of the transputer's address space, minus the internal RAM which overlays it (#80001000-#8003FFFF).

Note that most transputer compilers have storage allocation strategies which attempt to put frequently used data at a low memory address. This strategy will work well with the private SRAM as it is located just above the transputer's internal memory. However, in seeking ultimate performance, the user may wish to perform his own analysis of where code and data are placed.

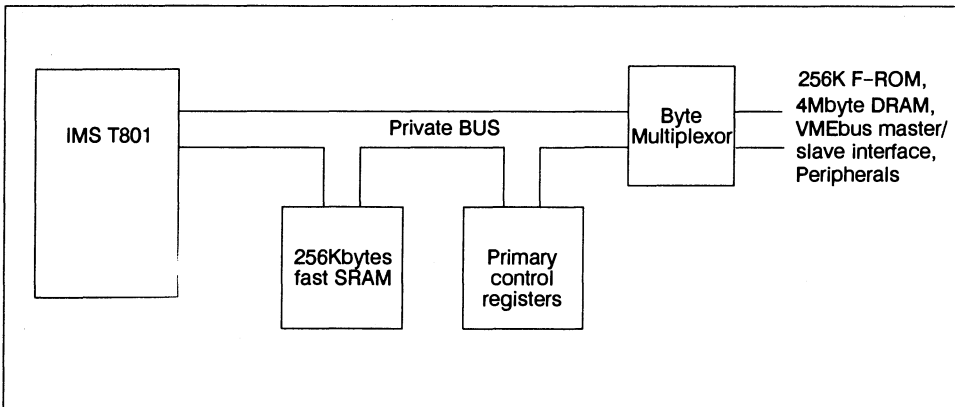


Figure 40.3 IMS T801 and private SRAM

40.2.2 Primary Control Registers

The primary control registers are only accessible by the IMS T801 and appear in two places in the memory map. The primary position is from #7FDC0000, while a secondary address at location zero is provided for compatibility with other transputer boards and TRAMs (table 40.1). This region of the address space is often required for VMEbus addressing and on the IMS B016 can be disabled by writing a one to the register as indicated in table 40.1.

The primary control registers are accessed as memory locations as shown in table 40.1. Bit 0 is the only active bit in these registers. The other bits are undefined when read and must be written as zero. These

1. The IMS B016 is available with either the 20MHz or 25MHz speed variant of the IMS T801 transputer.

registers contain the traditional "subsystem" registers as found on other transputer cards; control bits for the MAP-RAM and decode ram; enable control bits for VMEbus master and slave cycles and Byte Multiplexor control registers.

Address	Register
#xx00	Subsystem Reset/Error
#xx04	Subsystem Analyse
#xx40	Enable VMEbus Slave Access
#xx44	Enable Byte Multiplexor
#xx48	Next Cycle is the MAP RAM
#xx4C	Next cycle is the Slave Decode RAM
#xx50	Do not map these registers to address zero
#xx54	Enable VMEbus Master Accesses
#xx80-#xxFF	Byte Multiplexor Control Registers

Table 40.1 T801 Primary Control Registers (bit 0)

40.2.3 MAP-RAM

The MAP-RAM contains control information for every address which can be generated by the IMS B016's IMS T801 transputer. For every IMS T801 cycle, the MAP-RAM produces a set of control information which is used to determine how the various parts of the board's circuitry behave. For instance the MAP-RAM determines what kind of VMEbus cycle will be performed when the transputer attempts to perform a VMEbus master cycle. The MAP-RAM has an entry for each memory page. Because pages are one megabyte in size it is impossible to have different MAP-RAM entries for two addresses unless they lie on different megabyte pages.

The MAP-RAM needs to be initialised at system startup to contain the correct information for the application. Because of the way the MAP-RAM is written, it is unwise to attempt to change its contents after this point.

MAP-RAM entries are written as follows:

- 1 Write a "one" into the "Next Cycle is to MAP-RAM" control register.
- 2 The very next cycle must be a write to the address of the MAP-RAM page to be programmed. The data for the 12 MAP-RAM bits must be located in bits 4-15.

Note that the address written to is in fact the address to which the MAP-RAM entry refers. For instance, if you wished to set up the MAP-RAM entry for addresses #01000000-#01100000 (one megabyte), then you could write your MAP-RAM contents to any address in that range. For consistency, it is recommended that the first address in the page is always used.

The MAP-RAM controls the following aspects of the board's behavior:

- Whether the page is allocated as VMEbus address space.
- The VMEbus address modifier codes produced when the board preforms VMEbus master cycles.
- The value presented on the lower two VMEbus address lines during D08 and D16 cycles.
- The VMEbus data size.
- The VMEbus address space used for master transfers.
- The particular swap-function performed by the Byte Multiplexor for this page.

Bit	Function
4	Byte Multiplexor control bit 0
5	Byte Multiplexor control bit 1
6	Byte Multiplexor control bit 2
7	VMEbus Address Size control bit 0
8	VMEbus Address Size control bit 1
9	VMEbus Data Size control bit 0
10	VMEbus Data Size control bit 1
11	VMEbus address bit 0
12	VMEbus address bit 1
13	VIC Function Control Bit 1
14	VIC Function Control Bit 2
15	VMEbus Master Access Enable bit

Table 40.2 MAP RAM Control Bits (bits 4–15)

40.2.4 Dual-Access DRAM

A large memory, using fast DRAM devices, is accessible from both the IMS T801 and (with appropriate programming) from the VMEbus as slave memory. This kind of dual-access is sometimes called "dual-port memory" (although true dual-port memory is very expensive and about 100 times less dense than that used on the IMS B016).

Without any programming, the dual-access DRAM appears as the first four megabytes¹ of the transputer's address space, minus the internal RAM and private SRAM which overlay it (#80040000–#803FFFFF). The DRAM cycles in 160ns (four CPU cycles).

When accessed from the VMEbus as slave memory, the dual-access DRAM can occupy any VMEbus address in any address space (except A16). In addition, not all the dual-access DRAM need be accessible from the VMEbus. This feature is useful both when implementing a system which uses the A24 address space (which is only 16 megabytes) and when a secure multiprocessor system is desired. To derive the address in the dual-access DRAM which corresponds to a particular VMEbus address, ignore the top 10 address bits and treat the bottom 22 as an index into the four megabyte RAM. Note that this means that some "lost" memory which the IMS T801 can not see (because it is overlaid by the private SRAM) will be accessible from the VMEbus.

40.2.5 Byte Multiplexor

There are two main databusses on the board. One is local to the IMS T801, its SRAM and primary control registers. The other connects everything else including the dual-access DRAM and the VMEbus. Between these two 32-bit busses is a transceiver which has the ability to re-order bytes (see figure 40.4). This feature allows data to be moved at high-speed between little-endian processors (the IMS T801) and big-endian processors and peripherals (most VMEbus cards). The byte multiplexor can perform any byte reordering operation, the particular operation being programmable on a page-by-page basis using the MAP RAM. Up to eight reordering functions can be active at any time.

1. The IMS B016 is designed to allow versions to be manufactured with between one and sixteen megabytes of DRAM. The standard version has four megabytes.

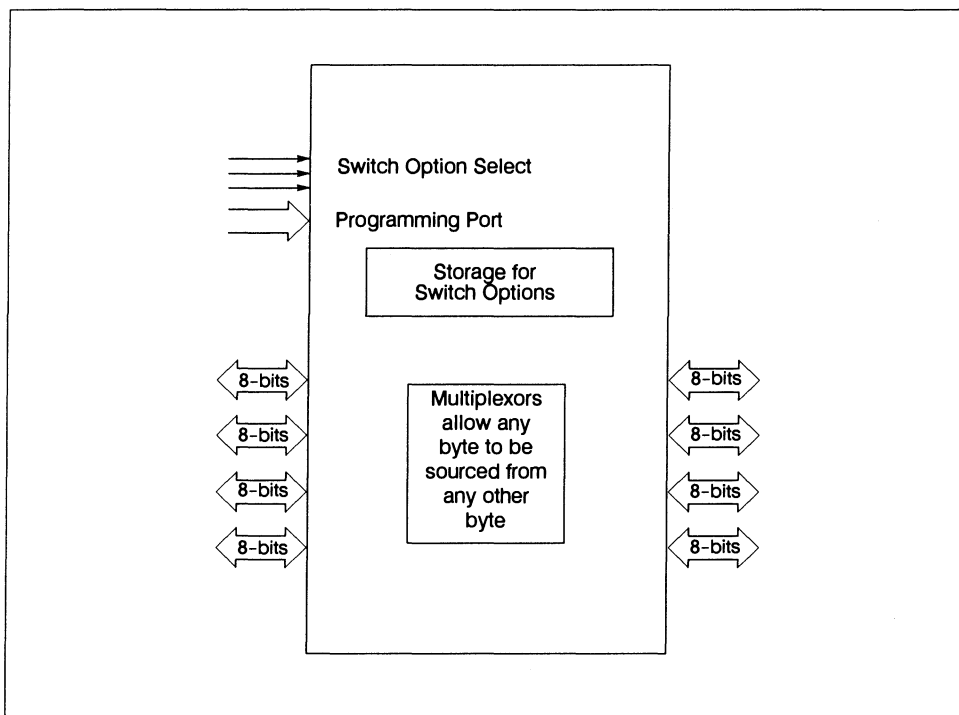


Figure 40.4 Byte Multiplexor

In order for the board to behave in a controlled manner at system startup, the operation of the Byte Multiplexor is disabled on reset and must be enabled via the "Enable Byte Multiplexor" register. When disabled, the Byte Multiplexor functions as a simple 32-bit transceiver.

The eight re-ordering functions can be re-programmed at any time by writing to the relevant registers. Note however that chaos is likely if the Byte Multiplexor is instructed to re-order memory where the currently executing program (or its data) resides. Note also that transputer byte-writes do not function correctly for regions of memory where byte multiplexing is in operation. This is because the four separate byte strobes cannot be multiplexed.

40.2.6 VME Features

The IMS B016 has a number of VMEbus master and slave features and can also act as a VMEbus Interrupter and as an Interrupt Handler.

Two distinct VMEbus slave areas are available. The first is the mailbox registers provided by the VMEbus control circuitry (VIC). These are only accessible in VMEbus address space A16 and are byte-wide. The address at which these registers appear is programmed via two hex switches. Note that the VIC must be programmed before the mailbox registers function.

The other slave is the dual-access DRAM. This is accessible from the VMEbus as a region of memory. The size and location of the decoded address region is controlled by the *VMEbus decode RAM* and by the VIC. This memory can occupy VMEbus address spaces A24 or A32 and can respond only to "supervisor" cycles if required. Note that the decoded address regions for the dual-access DRAM do not need to be contiguous or the same size as the available memory.

The dual-access DRAM responds correctly in slave mode to D08(E0), D16, D32, UAT, BLT, RMW and address-only cycles.

The IMS T801 can perform VMEbus master cycles simply by making memory accesses to certain regions of its address space. A certain amount of control register configuring is required before VMEbus master cycles can function. The MAP-RAM is used to distinguish between different VMEbus cycles at different address regions and has pages for each megabyte of T801 address space. The different VMEbus cycles selected by the MAP-RAM include D16(0-1)/D16(2-3) and the four different D08 cycles¹. Sometimes it may be desirable to access, say D16(0-1) and D16(2-3) words from the same VMEbus slave. For A24 and A16 slaves this is accomplished by programming the MAP-RAM with different configurations for the different access-types and using address regions which share the same lower 24 or 16-bits but have different upper bits. Thus the transputer will generate a full 32-bit (actually 30-bit) address, the upper bits allowing selection of different MAP-RAM entries, but the lower bits always selecting the same VMEbus A24 or A16 address. This scheme can not be used in the VMEbus A32 address space and users are limited to D32 cycles, D08 writes and *one* kind of D16 or D08 cycle.

VMEbus timeout is achieved by the slot 1 controller asserting Bus Error after some delay. Since the IMS T801 does not have a bus error pin, circuitry is provided to allow the bus error condition to cause an interrupt (event). When a bus error occurs, the IMS T801 will complete the outstanding memory cycle, any read data being corrupt; the user should ensure that the software handles bus errors in a controlled manner.

The IMS B016 has the capability to act as a VMEbus Interrupter and as an Interrupt Handler. Interrupts can be generated on all seven levels and the VIC can be programmed to recognise any or all of the seven levels. The IMS T801 lacks any vectored interrupt system and in the IMS B016, extra circuitry is provided to allow VMEbus interrupt vectors (status/ID) to be read from a status register.

5.2.7 F-ROM

The IMS T801 may boot from a link or from the on-board 256K ROM. These ROM devices are electrically reprogrammable *Flash* devices. Reprogramming can be achieved using the IMS T801, booted via a link² with the programming software. The devices are soldered to the board for extra reliability.

A programming tool, enabling the F-ROMs to be programmed with a binary code file generated from the INMOS toolset range of development tools, is available within the S514C device driver product. Full details of F-ROM programming are given in [1] and [2].

5.2.8 Serial Ports

Two full-duplex serial ports are available via a DUART device. These ports are buffered to RS-232 levels and connected to P2. The DUART can interrupt the IMS T801 on occurrences such as receiver ready and transmitter empty.

Spare input and output bits available on the DUART chip are used to create three extra "subsystem" ports. These signals are available on connector P2 (see table 40.15). Table 40.3 shows which DUART control bits correspond to which subsystem port. Note that these three extra subsystem ports control bits in the DUART using the logic level on the actual subsystem signals directly, rather than performing an inversion as in the "traditional" subsystem port.

The DUART can interrupt (Event) the IMS T801 by means of the VIC. It is connected to VIC local interrupt number 4 (LIQR4) and is active low.

Full information on the DUART and DUART programming is given in [3].

1. This scheme is used because, unlike 68020-type processors, the T801 lacks the lower two address pins and it is therefore impossible to determine which byte or word the processor requires.

2. For users interested in performing self-reprogramming, it is possible to have the card re-program itself. This is achieved by running a program in RAM (booted from ROM or link originally) which programs the ROMs at its convenience.

DUART Control Bit	Function
IP4	notAError
IP5	notBError
IP6	notCError
OP2	notAReset
OP3	notAAnalyse
OP4	notBReset
OP5	notBAnalyse
OP6	notCReset
OP7	notCAnalyse

Table 40.3 DUART - Three Extra Subsystem Ports

40.2.9 PEX Boards

Connector P3 provides a peripheral bus called "PEX". PEX boards are available from Radstone Technology Ltd. The PEX interface provides an 8-bit memory mapped address space of 256 locations. In addition, PEX daughterboards can cause local interrupts.

Details of the PEX interface will be provided in the full release version of the IMS B016 User Manual [1]. In the meantime, copies of the Radstone Technology documentation are available from INMOS.

40.2.10 Real-Time Clock

The RTC (Real-Time-Clock) provides a means for the board to keep track of the time of day and date, even when power is not applied to the main VMEbus power supply. This is achieved by using the VMEbus *Standby* power rail. For the RTC to retain the time when power is switched off, the user must supply 5v via the standby power rail. Some VMEbus card cages provide this facility either standard or as an option.

The RTC contains a total of 61 8-bit registers. These are addressable as byte 1 (bits 8-15) of the 32-bit words at addresses #7FD88000-#7FD88080. An indirection scheme using bits in control registers allows more registers to be addressed than are available in the memory map. A total of 33 bytes of non-volatile RAM are available in the RTC.

Full information on the Real Time Clock is given in the RTCDP8572 Datasheet (National Semiconductor) [4].

The RTC can interrupt (Event) the IMS T801 through the VIC. Its interrupt is connected to VIC local interrupt number 2 (LIRQ2) and is active low.

The RTC clock frequency is adjusted by means of the trimmer located adjacent to the DP8572 chip. Local temperature and supply variations mean that it may be necessary to recalibrate the clock. This is done by first using a frequency counter connected to pin 11 of the DP8572 and adjusting for 32.768KHz. Next, for exact adjustment, run the clock for a period and note whether it runs fast or slow. Make fine adjustments to achieve accurate timekeeping. Note that the clock may run at a slightly different frequency depending on whether it is supplied from the main or standby power supply. Most accurate timekeeping is achieved by calibrating the clock when it is powered from the most commonly used supply.

Address	Register
#7FD88000	Main status
#7FD88003	Real-time mode
#7FD88007	Output mode
#7FD8800C	Interrupt control 0/Periodic flag
#7FD88010	Interrupt control 1/Time save control
#7FD88013	100 th second counter
#7FD88017	Seconds counter
#7FD8801C	Minutes counter
#7FD88020	Hours counter
#7FD88023	Day of the month counter
#7FD88027	Months counter
#7FD8802C	Years counter
#7FD88030	Units julian counter
#7FD88033	100's julian counter
#7FD88037	Day of week counter
#7FD8804C	Seconds compare
#7FD88050	Minutes compare
#7FD88053	Hours compare
#7FD88057	Day of month compare
#7FD8805C	Months compare
#7FD88060	Day of week compare
#7FD88063	Seconds time save
#7FD88067	Minutes time save
#7FD8806C	Hours time save
#7FD88070	Day of month time save
#7FD88073	Months time save
#7FD88077	RAM
#7FD8807C	RAM/TEST

Table 40.4 Real-Time Clock Registers

40.2.11 Resets and Transputer System Services

The card circuitry is reset at power-on, following a VMEbus reset and by pushing the front panel reset button. All these actions create a *board reset*. In addition, the IMS T801 is reset at this time. The transputer can also be reset from the "ServicesUp" port on connector P2. This reset will not affect any other circuitry and allows the transputer to be re-booted without disturbing the rest of the card's circuitry.

Note that transputers do not typically clear their error signals on reset, a program which explicitly clears the error signal must be run. Transputer programs usually have this code included automatically. This means that the "error" LED remains lit after the card is reset.

The card's circuitry (including the IMS T801) can additionally be reset when a VMEbus master writes the appropriate data into the VIC mailbox registers. This feature allows a F-ROM booted card to be re-booted under the control of another VMEbus master card.

The IMS T801 is provided with a "subsystem" services port which is both software and electrically compatible with other INMOS cards. In addition, three extra electrically compatible subsystem ports are provided

for users who wish to control four independent sub-networks of transputers. These three extra ports were described earlier in section 40.2.8 are not software compatible with any other card.

40.2.12 The Front Panel

The IMS B016 front panel is shown in figure 40.5. A pushbutton resets all the card circuitry (including the IMS T801) and also produces a VMEbus reset if the card is configured as a slot 1 system controller.

The top (amber) LED lights when the IMS T801 error pin is asserted (see section 40.2.11).

The other three LEDs light respectively when the IMS T801 makes a VMEbus master access, when another VMEbus master makes a slave access to the IMS B016's dual-access DRAM and when the local bus is accessed by either the T801 or by the VMEbus. The brightness of the LEDs indicates the density of cycles being performed. Note that the LEDs are not intended to be balanced for brightness or calibrated with respect to each other.

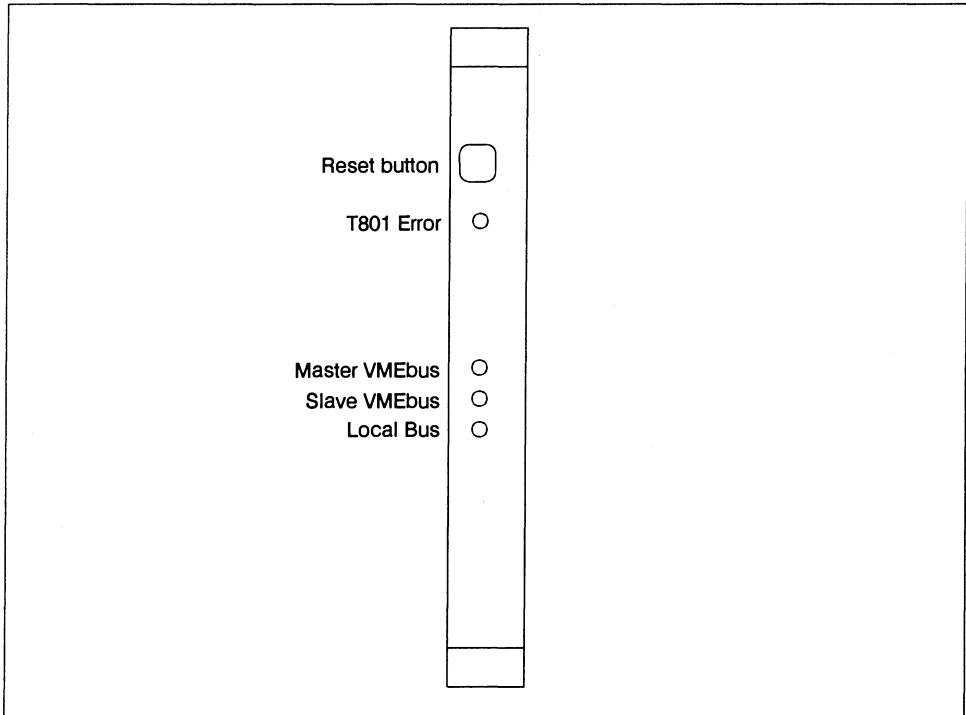


Figure 40.5 Front Panel

40.3 'Hard' Configuration Information

The IMS B016 is based around a VLSI controller and consequently there are far fewer configuration switches and jumpers than usually found on VMEbus cards. Some functions are however controlled by switches and jumpers (shown in figure 40.6).

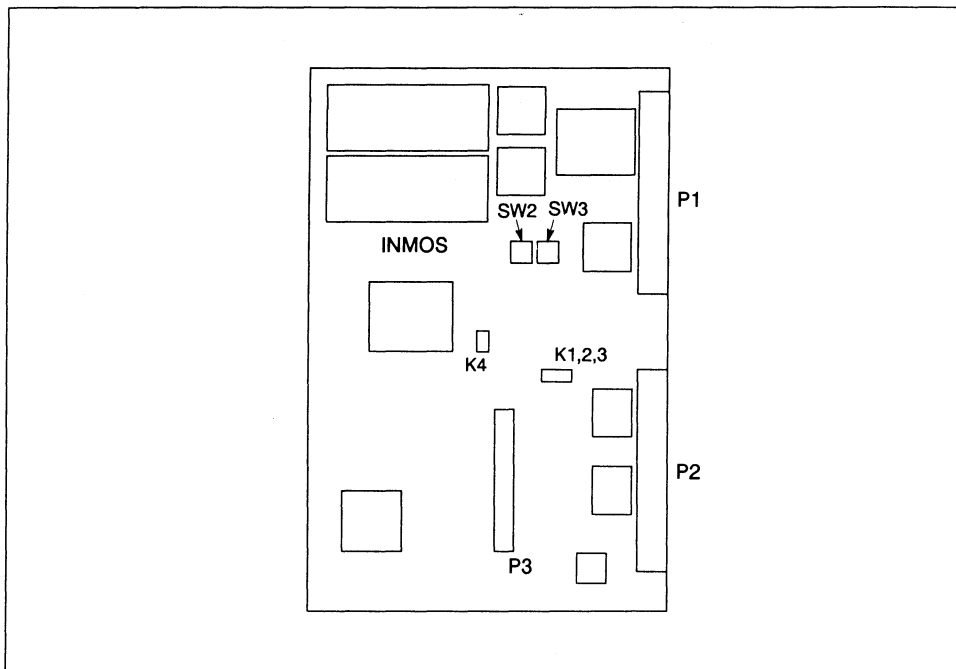


Figure 40.6 IMS B016 Connector and switch positions

The VMEbus address of the VIC VMEbus slave registers (mailboxes etc.) is selected by two hex switches (see figure 40.6). In setting the VMEbus address, a unique 8-bit binary number is selected which will be compared with the VMEbus addresses. The upper four bits of this number are set by SW2 while the lower four bits are set by SW3. A small screwdriver or trim-tool can be used to rotate the pointers on SW2,3 which must be rotated to the desired hex character.

The IMS T801's links can be set to work at two different speeds. For a link port on one device to talk successfully to another device, they must be set to the same speed. Current technology allows link speeds of 10 or 20Mbits/s. Jumper K2 sets the link speeds for all four of the IMS T801's links. When K2 is fitted, the links are set to 10Mbits/s, otherwise they are set to 20Mbits/s.

The IMS B016 can perform the VMEbus "Slot 1" controller functions. To enable this function, fit jumper K3. This should only be fitted if the board is installed in slot 1 of a VMEbus rack.

Jumper site K4 has three pins. The jumper can be fitted either over the center pin and the left pin, or over the center pin and the right pin. When programming the on-board FROMs, the programming voltage should be enabled by fitting K4 towards the INMOS logo. If the FROMs are not to be programmed, extra security of their contents can be achieved by fitting K4 towards the IMS T801. Table 40.5 summarises the jumper functions.

ID	Function
K1	IMS T801 Boots from link when fitted, otherwise from F-ROM.
K2	IMS T801 Links are 10Mbits/s when fitted, 20Mbits/s otherwise.
K3	When fitted, IMS B016 performs VMEbus slot 1 functions. Do not fit unless the IMS B016 is installed in slot 1 of the VMEbus card-cage.
K4	When not fitted F-ROM programming can not occur, it <i>must</i> be fitted for F-ROM programming to function.

Table 40.5 Jumper Functions

40.4 Memory Maps

Bit	Function
4	Byte Multiplexor control bit 0
5	Byte Multiplexor control bit 1
6	Byte Multiplexor control bit 2
7	VMEbus Address Size control bit 0
8	VMEbus Address Size control bit 1
9	VMEbus Data Size control bit 0
10	VMEbus Data Size control bit 1
11	VMEbus address bit 0
12	VMEbus address bit 1
13	VIC Function Control Bit 1
14	VIC Function Control Bit 2
15	VMEbus Master Access Enable bit

Table 40.6 MAP RAM Control Bits

Address	Function
#xx00	Subsystem Reset/Error register
#xx01	Subsystem Analyse register

Table 40.7 VMEbus Memory Map

Address	Function
#80400000-#FFFFFFF	VMEbus address space
#80040000-#803FFFFF	Dual-Access DRAM (4-cycles)
#80001000-#8003FFFF	Private SRAM (2-cycles)
#80000000-#80000FFF	On-Chip RAM (1-cycle)
#7FE00000-#7FFFFFFF	ROM Address Space
#7FDC0000-#7FDFFFFF	Control Registers
#7FD80000-#7FDBFFFF	Peripheral Address Space
#00040000-#7FCFFFFC	VMEbus Address Space
#00000000-#0003FFFC	(optional) Control Registers

Table 40.8 T801 Memory Map

Address	Register
#7FDB8000	PEX-Daughterboard
#7FD98000	Auxiliary Control Registers
#7FD90000	VIC Programming Registers
#7FD88000	Real-Time Clock
#7FD80000	DUART

Table 40.9 T801 Peripheral Address Map

Address	Function
#7FD90000	VMEbus Interrupter Interrupt Control
#7FD90004-#7FD90001C	VMEbus Interrupter Control 1-7
#7FD90020	DMA Status Interrupt Control
#7FD90024-#7FD90002C	Local Interrupt Control 1-7
#7FD90040	ICGS Interrupt Control
#7FD90044	ICMS Interrupt Control
#7FD90048	Error Group Interrupt Control
#7FD9004C	ICGS Interrupt Vector
#7FD90050	ICMS Interrupt Vector
#7FD90054	Local Interrupt Vector
#7FD90058	Error Group Vector
#7FD9005C	Interprocessor Comms. switch
#7FD90060-#7FD9007C	Interprocessor Comms. 1-7
#7FD90080	VMEbus interrupt request and status
#7FD90084-#7FD9009C	VMEbus interrupt vectors 1-7
#7FD900A0	Transfer timeout register
#7FD900A4	Local bus timing
#7FD900A8	Block transfer definition
#7FD900AC	VMEbus interface configuration 1
#7FD900B0	Arbiter and requester configuration
#7FD900B4	Address modifier source
#7FD900B8	Bus error status
#7FD900BC	DMA status (not used)
#7FD900C0	Slave select 0 control 0 (not used)
#7FD900C4	Slave select 0 control 1 (not used)
#7FD900C8	Slave select 1 control 0
#7FD900CC	Slave select 1 control 1
#7FD900D0	Release control
#7FD900D4	Block transfer control
#7FD900D8	Block transfer length 0
#7FD900DC	Block transfer length 1
#7FD900E0	System Reset

Table 40.10 VIC Register Memory Map (bits 0-7)

Address	Register
#xx00	Subsystem Reset/Error
#xx04	Subsystem Analyse
#xx40	Enable VMEbus Slave Accesses
#xx44	Enable Byte Multiplexor
#xx48	Next Cycle is to the MAP RAM
#xx4C	Next Cycle is to the Slave Decode RAM
#xx50	Do not MAP these registers to address zero
#xx54	Enable VMEbus Master Accesses
#xx80-#xxFF	Byte Multiplexor Control Registers

Table 40.11 T801 Primary Control Registers (bit 0)

Address	Register
#7FD98000	Clear BusError Interrupt
#7FD98004	Clear Event
#7FD98008	Enable Vpp to F-ROMs
#7FD9800C	Read Vpp voltage sensor
#7FDA0000	Read VIC Interrupt Vector

Table 40.12 T801 Auxiliary Registers (bit 8)

40.5 Specification

40.5.1 Mechanical and Thermal Details

The IMS B016 is designed to accord with DIN 41494 and IEC 297 standards. The board is nominally 160mm by 233.35mm. The supplied front panel width is 4HP (approximately 20mm). This is compatible with a board-to-board pitch in a card cage of 0.8in.. M2.5 fastening bolts are provided on the front panel, mating with tapped holes in the card cage to fix the board securely. Front panel handles allow the board to be removed from the card cage (by un-screwing the retaining bolts and pulling hard on the handles). Note that the front panel is *required* when operating the IMS B016 in a card cage, both for mechanical rigidity and to give correct cooling air flow.

No components protrude more than 2.47mm below the surface of the board. To fit in a 0.8in. pitch card-cage, components should not protrude more than 13.7mm above the surface of the board.

Adequate cooling air flow must be provided to maintain the components on the board within their operating temperature. High reliability should not be expected from boards not provided with adequate cooling. Air flow should run parallel to the board surface and parallel to the front panel. The IMS B016 dissipates 25W maximum which means that a J1/J2 backplane must be used. The cooling air flow required for a particular application will need to be determined empirically.

The IMS B016 can be used as a slave in a Sun workstation, using a CA12 card-frame adapter to connect to the VMEbus.

	Operating	Storage
Temperature	0 to +50°C ambient air	-55 to +85°C
Relative Humidity	95% non condensing	95% non condensing
Thermal shock	<0.08°C/s	<0.15°C/s
Altitude	-300 to +3000m	-300 to +16000m

Table 40.13 Environmental Details

40.5.2 Electrical Details

The IMS B016 requires power supply voltages in accordance with the VMEbus specification. The +5V DC supply must be between 4.875V and 5.25V and have less than 50mV peak-peak noise and ripple between DC and 10MHz. The maximum power consumption of the IMS B016 is 25W. The IMS B016 does not incorporate protection against incorrect power supplies. Major damage can result from operating the board outside its power supply range.

40.5.3 VMEbus capability

For easy description of VMEbus boards, the VMEbus specification defines a number of "capability" abbreviations. The capabilities applying to the IMS B016 are listed below:

- 1 A32/A24/A16:D32/D16/D08(EO) UAT,BLT,RMW SLAVE
- 2 A32/A24/A16:D32/D16/D08(EO) BLT MASTER
- 3 INT(1-7):D08(O) INTERRUPTER
- 4 INT(1-7):D32 INTERRUPT HANDLER
- 5 Full Slot 1 Bus controller, PRI and RRS arbiter
- 6 ROR, RWD and Fairness bus requester
- 7 6U high-double height board

40.6 Connector Pin Assignments

The connector assignments for P1, P2 and P3 are shown as they appear when looking at the connectors. This is not in strict alphanumeric order.

Pin	row c	row b	row a
1.	D08	BBSY*	D00
2	D09	BCLR*	D01
3	D10	ACFAIL*	D02
4	D11	BG0IN*	D03
5	D12	BG0OUT*	D04
6	D13	BG1IN*	D05
7	D14	BG1OUT*	D06
8	D15	BG2IN*	D07
9	GND	BG2OUT*	GND
10	SYSFAIL*	BG3IN*	SYSCLK
11	BERR*	BG3OUT*	GND
12	SYSRESET*	BR0*	DS1*
13	LWORD*	BR1*	DS0*
14	AM5	BR2*	WRITE*
15	A23	BR3*	GND
16	A22	AM0	DTACK*
17	A21	AM1	GND
18	A20	AM2	AS*
19	A19	AM3	GND
20	A18	GND	IACK*
21	A17	SERCLK	IACKIN*
22	A16	SERDAT	IACKOUT*
23	A15	GND	AM4
24	A14	IRQ7*	A07
25	A13	IRQ6*	A06
26	A12	IRQ5*	A05
27	A11	IRQ4*	A04
28	A10	IRQ3*	A03
29	A09	IRQ2*	A02
30	A08	IRQ1*	A01
31	+12V	+5VSTDBY	-12V
32	+5V	+5V	+5V

Table 40.14 Connector P1 pin assignments

Pin	row c	row b	row a
1	TxA	VCC	TxB
2	RxA	GND	RxB
3	OpA	RESERVED (nc)	OpB
4	IpA	A24	IpB
5	GND	A25	GND
6	nc	A26	nc
7	P2Link0Out	A27	P2Link1Out
8	P2Link0In	A28	P2Link1In
9	GND	A29	GND
10	GND	A30	GND
11	nc	A31	nc
12	P2Link2Out	GND	P2Link3Out
13	P2Link2In	VCC	P2Link3In
14	GND	D16	GND
15	nc	D17	nc
16	notAReset	D18	notBReset
17	notAAnalyse	D19	notBAnalyse
18	notAError	D20	notBError
19	GND	D21	GND
20	GND	D22	GND
21	nc	D23	nc
22	notCReset	GND	notSubReset
23	notCAnalyse	D24	notSubAnalyse
24	notCError	D25	notSubError
25	GND	D26	GND
26	GND	D27	GND
27	nc	D28	nc
28	notUpReset	D29	notDownReset
29	notUpAnalyse	D30	notDownAnalyse
30	notUpError	D31	notDownError
31	GND	GND	GND
32	GND	VCC	GND

Table 40.15 Connector P2 pin assignments

Pin	Signal	Pin	Signal
1	+12v	21	VCC
2	Vcc	22	VALWAYS
3	A7	23	PEXSEL*
4	A6	24	DSACK0*
5	A5	25	PROCLK*
6	A4	26	BERR*
7	A3	27	Reserved
8	A2	28	PEXINT*
9	A1	29	PEXRES*
10	A0	30	PEXIN*
11	GND	31	PEXOUT*
12	GND	32	GND
13	D7	33	GND
14	D6	34	SYSCLK
15	D5	35	R/W*
16	D4	36	AS*
17	D3	37	DS*
18	D2	38	PEXPRES*
19	D1	39	VCC
20	D0	40	-12v

Table 40.16 Connector P3 pin assignments

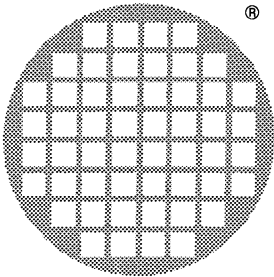
40.7 References

- 1 *IMS B016 User Manual*, INMOS, 1990
- 2 *Data Sheet and Application Notes for 28F512 F-ROM*, INMOS
- 3 *SCN2681 Datasheet*, Philips/Signetics
- 4 *DP8572 Datasheet*, National Semiconductor

40.8 Ordering Information

Description	Order number
IMS B016-1 VMEbus Master/Slave Board 25MHz operation	IMS B016-1
IMS B016-2 VMEbus Master/Slave Board 20MHz operation	IMS B016-2
Associated products	
IMS S514 Device driver for Sun Workstation	IMS S514
IMS CA12 Card frame adapter for Sun Workstation	IMS CA12

Table 40.17 Ordering Information

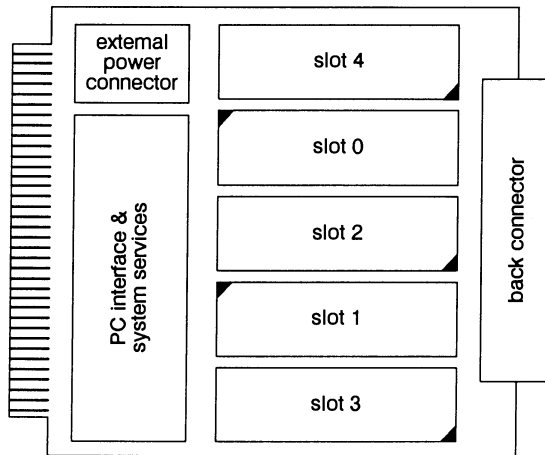


inmos®

IMS B015

NEC 9800 series PC board

Engineering Data



FEATURES

- 5 slots for INMOS TRAMs (Transputer Modules)
- INMOS link adaptor interface to the NEC expansion bus
- Interrupt capability
- Choice of IO address
- Conforms to INMOS Module-Motherboard Architecture (*INMOS Technical Note 49*)
- Can be used as an interface to external transputer systems

GENERAL DESCRIPTION

The IMS B015 is a motherboard for *Transputer Modules* (TRAMs) for the NEC PC-9800 series of personal computers. It allows transputer modules to be fitted to a 9800 series PC for program development, and application acceleration.

The IMS B015 has five slots for TRAMs and an interface to the 9800 series PC expansion bus. This allows the PC to communicate with and reset the TRAMs. It also has connections which allow it to connect to other transputers, TRAMs, or transputer boards (such as another IMS B015).

41.1 Description

41.1.1 Link connections

The INMOS link connections on the IMS B015 are arranged as shown in figure 41.1. Two links from each TRAM slot are taken to the back connector (P1). Slots 1–4 are connected in a pipe-line. Some of the link connections can be configured by jumper blocks K1 and K2. K1 connects the PC interface link to SLOT0 link0 or connects both links to P1. K2 connects SLOT0 link2 to SLOT1 link1 or connects both links to P1.

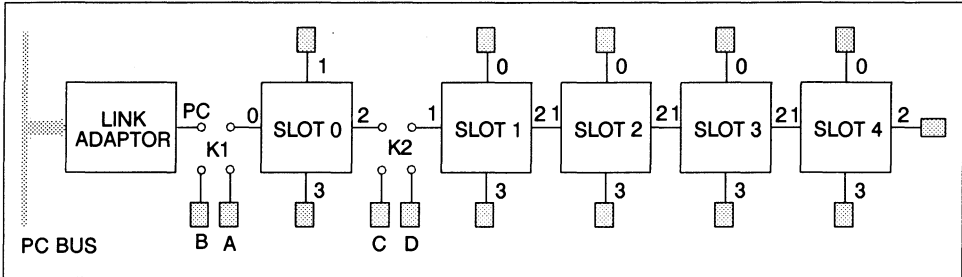


Figure 41.1 Link connections

Thus, all of the TRAMs can be connected in a pipeline or all of the links from *slot0* can be taken to P1. The PC interface link and *slot1 link1* can also be taken to P1.

link A in	link A out
slot0 link0 in	slot0 link0 out
PC link out	PC link in
link B out	link B in

Table 41.1 K1 signals

link C out	link C in
slot0 link2 out	slot0 link2 in
slot1 link1 in	slot1 link1 out
link D in	link D out

Table 41.2 K2 signals

41.1.2 Link speed selection

TRAMs have two link speed select pins: **LinkSpeedA** and **LinkSpeedB**. On the IMS B015, there are five link speed select jumpers. These control the TRAM link speeds as shown in table 41.3. To determine the effect of these jumpers on the link speeds of the TRAMs you are using, refer to the data sheets for the TRAMs.

Jumper	Controls	Inserted	removed
J9	PC link	10Mbits/s	20Mbits/s
J14	slot0 LinkSpeedA	0	1
J13	slot0 LinkSpeedB	0	1
J11	slot1-4 LinkSpeedA	0	1
J12	slot1-4 LinkSpeedB	0	1

Table 41.3 IMS B015 link speed selection

41.1.3 System Services

A TRAM has a **reset** input, an **analyse** input, and a **notError** output. *System services* is used as a collective term for these signals. The system services signals on the IMS B015 are arranged as shown in figure 41.2.

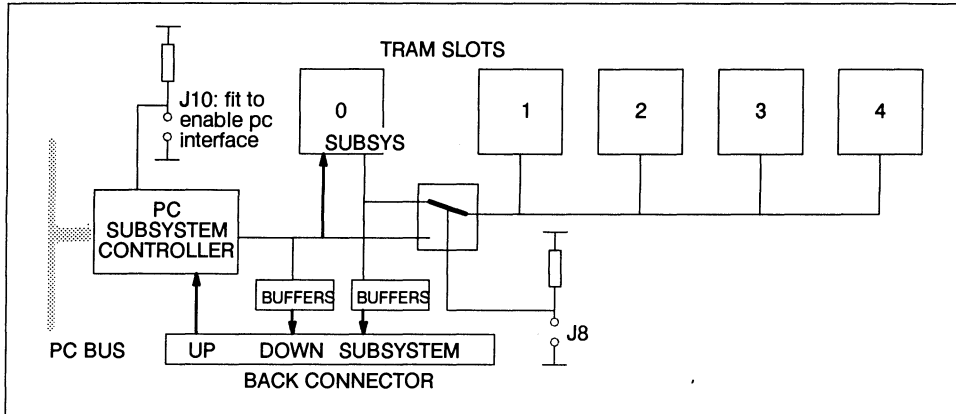


Figure 41.2 System services

If the PC interface is enabled, system services for slot 0 come from the PC interface. If the PC interface is disabled, system services for slot 0 come from the **Up** port.

Some types of TRAM have three extra pins which allow them to drive reset and analyse signals and to monitor an error signal. This is termed *sub-system control*. A TRAM with sub-system control can reset, analyse, and monitor the error signals of other TRAMs. If the TRAM in slot0 of the IMS B015 has sub-system control, the IMS B015 can be configured so that this TRAM controls the TRAMs in slots 1–4. The sub-system control signals from slot 0 are also available at the **SubSystem** port on the back connector of the IMS B015. This allows the TRAM in slot 0 to control TRAMs on other boards.

TRAMs in slots 1–4 can be controlled by the same system services signals as the TRAM in slot0; or they can be controlled by the slot 0 sub-system. The selection is made by a single jumper.

41.1.4 Up, Down, and Subsystem

The IMS B015 has three system services ports on the back connector. These are called *Up*, *Down*, and *Subsystem*. Each of these ports comprises a reset, analyse, and error signal.

The Up port allows other TRAMs or transputer boards to control the IMS B015. It can be connected directly to the Down or Subsystem port of another transputer board. The PC interface must be disabled to make use of the **Up** port.

The Down port repeats the reset and analyse signals from the Up port if the PC interface is disabled. If the PC interface is enabled, it repeats the reset and analyse signals from the PC interface. The error signal from the Down port is reported to the PC interface if the PC interface is enabled, and to the Up port if the PC interface is disabled. The Down port can be connected directly to the Up port of another transputer board (such as another IMS B015).

The SubSystem port allows a TRAM with a sub-system controller in slot 0 of the IMS B015 to control other transputer boards. The subsystem port can be connected directly to the Up port of another transputer board.

41.1.5 PC interface

The IMS B015 is designed to plug into any NEC 9800 series PC. It occupies one expansion slot. The interface to the PC expansion bus allows the PC to

- reset the transputer modules
- load code onto the TRAMs
- test the combined error signal from the TRAMs
- analyse the TRAM network to identify the cause of an error
- communicate with the TRAM network

The interface can be polled or interrupt driven.

41.1.6 IO Address

The IMS B015 can be placed at one of several IO addresses in the PC's IO address space. The PC interface can also be disabled so that it does not respond to any address. Table 41.4 shows which jumpers should be fitted to enable the IMS B015 at a particular IO address. The IO addresses are in hexadecimal (indicated by a #). A jumper must be fitted if there is a * in the column, otherwise it must be removed, x means that it does not matter if a jumper is fitted or not fitted. The IMS B015 occupies a block of sixteen addresses starting at the base address.

Base address	J6	J7	J10
#0D0	*	*	*
#1D0	*		*
#2D0		*	*
#3D0			*
none	x	x	

Table 41.4 IO address

41.1.7 Reset, Analyse and Error registers

These registers allow software running on the PC to control the TRAMs on the IMS B015 and to monitor their combined error status. Their offsets from the board base address are given in table 41.5.

Register	Address	Read/Write	Asserted State
reset	base + #8	write only	1
analyse	base + #A	write only	1
error	base + #8	read only	0

Table 41.5 PC system services register locations

Writing 1 to the *reset* register asserts reset to the TRAM in slot 0 and asserts **notDownReset**. Writing 1 to the *analyse* register asserts analyse to the TRAM in slot 0 and asserts **notDownAnalyse**. The *error register* indicates whether any of the TRAMs has asserted its error flag or if **notDownError** is asserted. A 0 is read if any of the TRAMs has asserted its error flag.

41.1.8 Interface link

To allow the PC to load code to the TRAM network, an interface from the PC expansion bus to an INMOS link is provided. The interface uses an IMS C012 link adaptor. This device is like a UART: it has *output data* and *input data* registers, and *output status* and *input status* registers. These are located at the addresses shown in table 41.6.

Register	Address	Read/Write
input data	base + #0	read only
output data	base + #2	write only
input status	base + #4	read/write
output status	base + #6	read/write

Table 41.6 PC interface link register locations

The output status register contains an *output ready* bit which is 1 if the output data register is empty. The interrupt enable bit allows the link adaptor to assert one of the NEC PC's interrupt lines when the output data register is empty.

Bit	Name	Function
0	output ready	0 if output data register is busy
1	interrupt enable	1 to enable output interrupt
2-7		none

Table 41.7 Output Status Register

The input status register contains a *data present* bit which will be 1 if the input data register contains a valid byte received from the link. The interrupt enable bit allows the link adaptor to assert one of the NEC PC's interrupt lines when the input data register contains data.

Bit	Name	Function
0	data present	1 if data has been received
1	interrupt enable	1 to enable input interrupts
2-7		none

Table 41.8 Input Status Register

Interrupts

The link adaptor can be made to interrupt the NEC PC when it has received or transmitted a byte. Interrupt on output and interrupt on input can be enabled and disabled separately but both conditions drive the same signal to the PC bus. The interrupt signal can be connected to either IR31, IR61 or IR121. These correspond to channels in the PC's programmable interrupt controllers as shown in table 41.9.

Interrupt line	PIC channel	Interrupt Vector	Fit Jumper
IR31	MPIC-IR3	#0B	J5
IR61	MPIC-IR6	#0E	J4
IR121	SPIC-IR4	#14	J3

Table 41.9 Interrupt lines

Only one interrupt line should be driven at any time so only one of J3, J4, J5 should be fitted. If it is not desired to use interrupts, all of the jumpers should be removed.

41.1.9 External power supplies

The NEC PCs can supply a maximum of 0.5A to an expansion board. Because an IMS B015 when populated with TRAMS can require more than this, there is an option to supply power to the IMS B015 from an external power supply while remaining connected to the NEC expansion bus.

The socket for external power (P2) is the same type and pinout as a disk drive power connector. The pinout is given in table 41.10. If you wish to supply power to the IMS B015 from an external supply

- 1 REMOVE J1 and J2.
- 2 Connect a suitable 5V power supply to P2.
- 3 Insert the IMS B015 into the NEC PC.
- 4 Switch on the PC.
- 5 Switch on the power to the IMS B015.
- 6 ALWAYS turn on the IMS B015 power last and turn off the IMS B015 first.

Pin	Signal
1	
2	0V
3	0V
4	5V

Table 41.10 P2 connector pinout

The IMS B015 and any TRAMs will then draw all of their power from the external supply but is interfaced to the PC as before. Note that the 0V of the external power supply is connected to the NEC PC 0V. P2 is also suitable for supplying power to the IMS B015 when it is not connected to a PC.

41.2 External Connections

pin	row c	row b	row a
1	GND	GND	GND
2	nc	nc	nc
3	link A out	slot2 link0 out	slot4 link0 out
4	link A in	slot2 link0 in	slot4 link0 in
5	GND	GND	GND
6	GND	GND	GND
7	nc	nc	nc
8	slot0 link1 out	slot2 link3 out	slot4 link3 out
9	slot0 link1 in	slot2 link3 in	slot4 link3 in
10	GND	GND	GND
11	GND	GND	GND
12	nc	nc	nc
13	link C out	slot1 link0 out	slot3 link0 out
14	link C in	slot1 link0 in	slot3 link0 in
15	GND	GND	GND
16	GND	GND	GND
17	nc	nc	nc
18	slot0 link3 out	slot1 link3 out	slot3 link3 out
19	slot0 link3 in	slot1 link3 in	slot3 link3 in
20	GND	GND	GND
21	GND	GND	GND
22	nc	nc	nc
23	link B out	link D out	slot4 link2 out
24	link B in	link D in	slot4 link2 in
25	GND	GND	GND
26	nc	nc	nc
27	nc	nc	nc
28	notUpReset	notSubsystemReset	notDownReset
29	notUpAnalyse	notSubsystemAnalyse	notDownAnalyse
30	notUpError	notSubsystemError	notDownError
31	GND	GND	GND
32	GND	GND	GND

Table 41.11 IMS B015 back connector pinout

41.3 Specification

feature		Unit	Notes
TRAM slots	5		
Interface type	IMS C012 link adaptor to NEC PC expansion bus		
Length	6.65	inch	1
Width	5.85	inch	
Component height above PCB	12.0	mm	
Component height below PCB	4.0	mm	
Weight	150	g	
Storage temperature	0-70	°C	
Operating temperature	0-50	°C	
Power supply voltage (Vcc)	4.75-5.25	Volt	
Power consumption	1.7(typ.) 2.1(max.)	W	

Table 41.12 IMS B015 specification

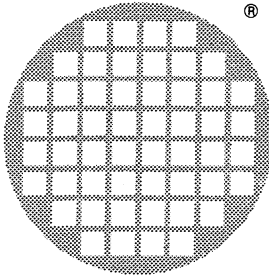
Notes:

- 1 This dimension includes the PCB thickness of 1.6mm.

41.4 Ordering Information

Description	Order Number
IMS B015 Module Motherboard	IMS B015-1

Table 41.13 Ordering information



inmos®

IMS B012

Double extended eurocard

Engineering Data

FEATURES

- 16 transputer module (TRAM) slots
- IMS T222 – 16 bit transputer
- Two IMS C004 programmable 32 way switches
- Double Extended Eurocard
- 40 links available at edge connectors

GENERAL DESCRIPTION

The IMS B012 is a eurocard TRAM motherboard designed to fit into standard card cages for double extended eurocards. The IMS B012 provides 16 slots for TRAMs, with IMS C004s to provide a wide variety of configurations. A possible application might be an image or speech recognition system using two B420s (vector processing TRAMs) for feature extraction, an IMS B427 (8Mbyte TRAM) running LISP or another AI language for recognition, and an IMS B419 providing graphical output, all on one board.

The IMS B012 can also be used to provide switchable backplane connectors for other boards plugged into a backplane.

42.1 IMS B012 Double Eurocard Motherboard engineering data

42.1.1 Introduction

The IMS B012 is a eurocard TRAM motherboard which is designed to fit into standard 6U, 220mm deep, DIN41494 (and IEC 297) card cages. It has slots for up to 16 TRAMs. These are transputer-based circuit modules which communicate with the outside world by means of INMOS serial links (a link is a two-wire serial communications port which can run at up to 20 MHz). The smallest TRAM is 'size 1'. Each of the 16 sites for modules on the IMS B012 will accept a size 1 module. Each module site, or 'slot' has connections for four INMOS links which are designated *link 0*, *link 1*, *link 2* and *link 3*. TRAMs which are larger than size 1 can be mounted on the B012. A larger module occupies more than one slot and need not use all of the available link connections provided by the slots which it occupies.

The B012 has two IMS C004 link switch ICs. These devices are able to connect together links from the slots and 32 links which are available on an edge connector. The connections can be changed by control data passed to the board down a configuration link, which may come from some master system or from one of the TRAMs on the B012 itself.

42.1.2 Hardware Description

The 16 module sites or slots provided by the IMS B012 are 16-pin sockets in accordance with the TRAM Specification (INMOS Technical Note 29). The slots are numbered as shown on the board silk screen and in figure 42.1.

The IMS B012 has two DIN41612 96-way edge connectors, P1 and P2. These carry almost all signals and power to/from the board and are easily identified from the board silk screen printing and from figure 42.1. P2 carries power, pipeline and configuration links and system control signals (reset and analyse and error).

NOTE — it is very important that you do not mix up P1 and P2. Unrecoverable damage to the IMS B012 will almost certainly result.

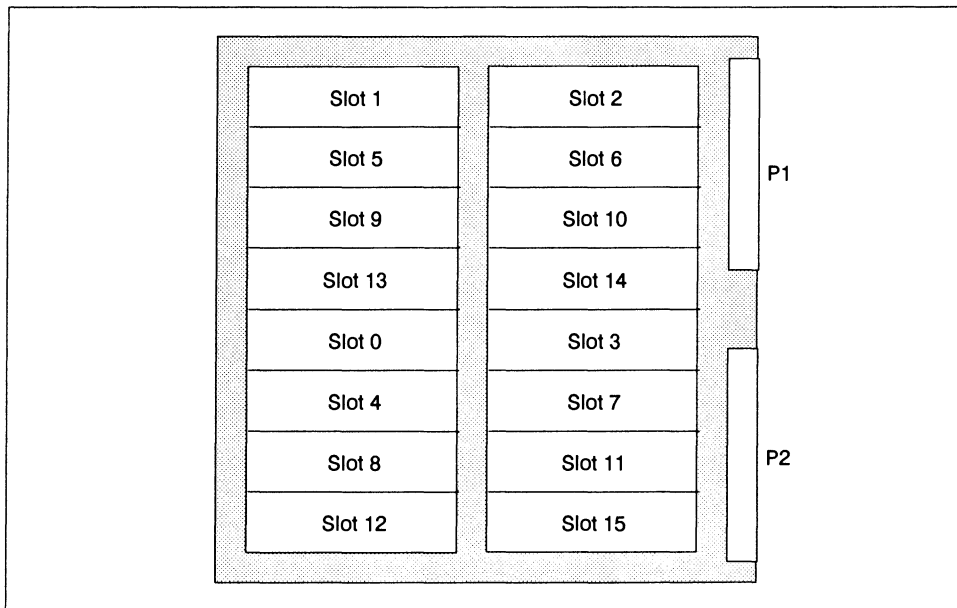


Figure 42.1 IMS B012 slot positions

Link Connections

The link connections to the 16 slots are organised as follows: Two links from each slot (links 1 and 2) are used to connect the 16 slots as a 16-stage pipeline (in a pipeline, multiple processors are connected end-

to-end as in figure 42.2). The pipeline is actually broken by jumper block K1. K1 will usually be jumpered in the standard way to give a 16-stage pipeline but can allow other combinations.

When modules larger than size 1 are used, the pipeline will be broken at the slots which are underneath large modules. Special plugs, called pipe-jumpers are provided (figure 42.3 shows a pipe jumper). These plug into the unused slot and connect the signals for links 1 and 2 together, thus connecting the pipeline through to the next TRAM in the chain.

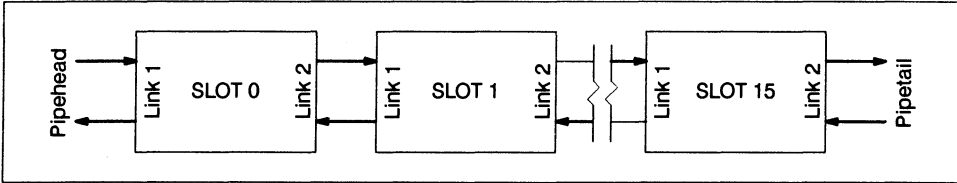


Figure 42.2 A module pipeline

Link 1 on slot 0 is wired to an edge connector (P2) and is called *PipeHead*. Link 2 on slot 15 is also taken to P2 and is called *PipeTail*. By connecting the pipe heads and tails from multiple boards together, a large, multi-board pipeline is created.

The other two links (links 0 and 3) of each slot are, in general, connected to two IMS C004 programmable link switches (For detailed information on the IMS C004 see the *IMS C004 Link Switch Data Sheet*).

The IMS C004 has 32 input pins and 32 output pins, plus an INMOS link (ConfigLink) used to send configuration information to the IMS C004. Any of the output pins can be 'connected' to any of the input pins, so a signal presented on the input pin would be buffered and transmitted on the output pin (with a slight delay).

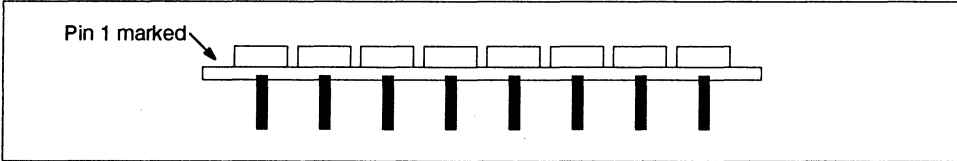


Figure 42.3 A Pipe-Jumper

The switch connections are made according to information sent to the IMS C004 down its ConfigLink. The two IMS C004s on the IMS B012 allow 64 link connections to be made under software control.

In most applications using the IMS C004, the device is treated as a 32-way Link Crossbar. This means that 32 INMOS links, each of which has two signals, may be connected to each other in an arbitrary fashion. That is to say that any of the 32 links can be 'connected' via the IMS C004 to any of the other 31 links. The IMS B012 uses the IMS C004s in a slightly different way, the difference being that the two signals from any particular link are routed through *different* IMS C004 devices. So if the LinkIn signal comes from one IMS C004, then the LinkOut signal will go to the other IMS C004. Figure 42.4 shows the general routing of link signals.

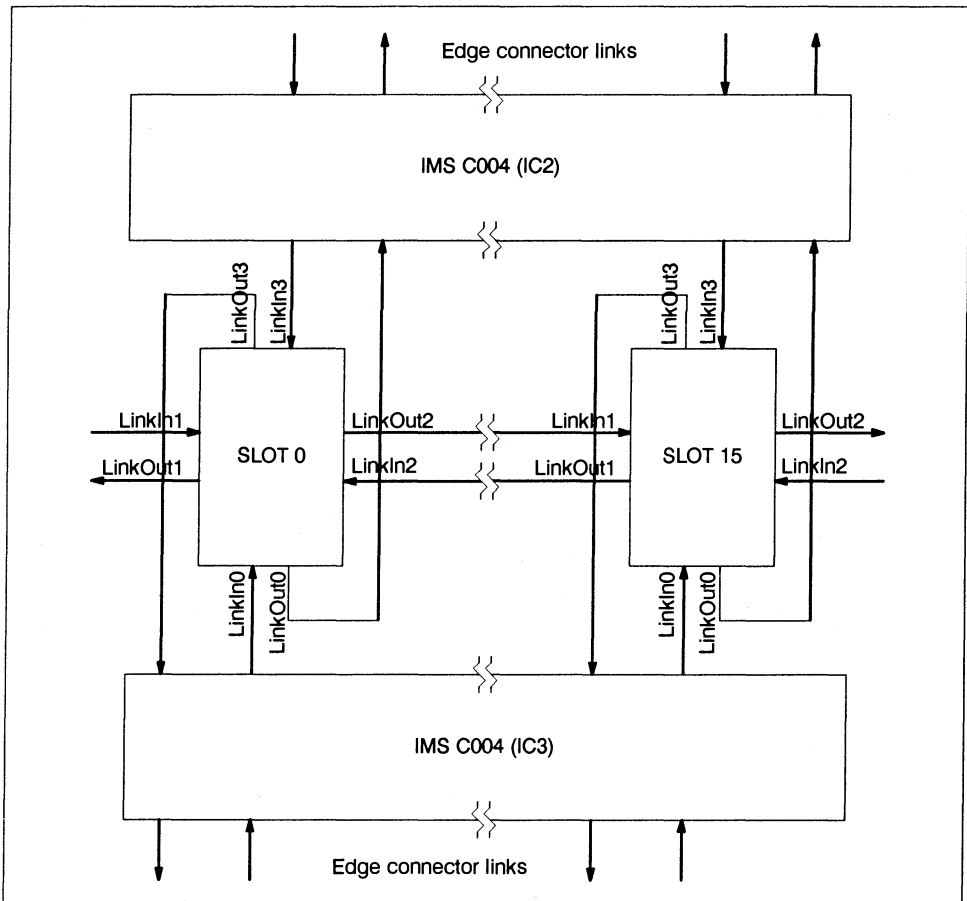


Figure 42.4 Link Organisation – slots to IMS C004s

The link output signals from all the link 0s on all the slots (16 signals) are connected to 16 inputs of one IMS C004 (IC2). The link input signals from all the link 3s on all the slots (16 signals) are connected to 16 outputs of the same IMS C004. The remaining 16 inputs and 16 outputs of that IMS C004 are connected to an edge connector (P1).

The other IMS C004 (IC3) is connected similarly, except that 16 of its inputs are connected to the outputs of all link 3s on all the slots, and 16 of its outputs are connected to the inputs of all link 0s on all the slots. The remaining inputs and outputs are connected to P1.

The result of this connection scheme is that any link 0 on any module may be routed via the IMS C004s to any link 3 on any module, but may not be routed to any of the link 0s on any other module. The same is true for link 3s on any modules, they may not be routed to any other link 3. Each of the links 0 and 3 on any module may be routed to any of half of the link connections on edge connector P1 (see below).

By hardwiring two of the edge connector links together off the board, any of the slot link 0s can be routed to another slot link 0, via the two connected edge links.

MMS (Module Motherboard System) software which is available for all module motherboards allows the configuration of module interconnection to be achieved easily from a connection list description of the desired network.

Slot 0 link 0 is not directly connected to its appropriate IMS C004 pins. It is connected to edge connector P2, along with the respective pins from the IMS C004s. A link jumper connector which is supplied with the board can be used to make the connection between slot 0 link 0 and the IMS C004s. slot 0 link 0 is taken to P2 in order to provide two links which are directly connected to module 0 on an edge connector. For some applications it will be useful to by-pass the IMS C004 switches in this way.

Similarly slot 0 link 3 is connected to pins on jumper-block K1. Usually K1 will be configured to connect slot 0 link 3 to the appropriate pins on the two IMS C004s. Because there are K1 pins which are connected to pins on edge connector P2, slot 0 link 3 can be wired to the edge connector instead of to the IMS C004s. It is possible, using a non-standard configuration of K1, to take links 0, 1 and 3 from slot 0 off the board via P2. This is useful if slot 0 contains a TRAM which is controlling a system of other TRAMs or transputers.

Figure 42.5 shows the organisation of the pipeline links and the links which are available on P2 and K1.

IMS C004 link switches introduce a small delay into the signals which they switch. If multiple IMS C004s are introduced into a link signal path, as with multi-board IMS B012 systems, the link data rate may be reduced.

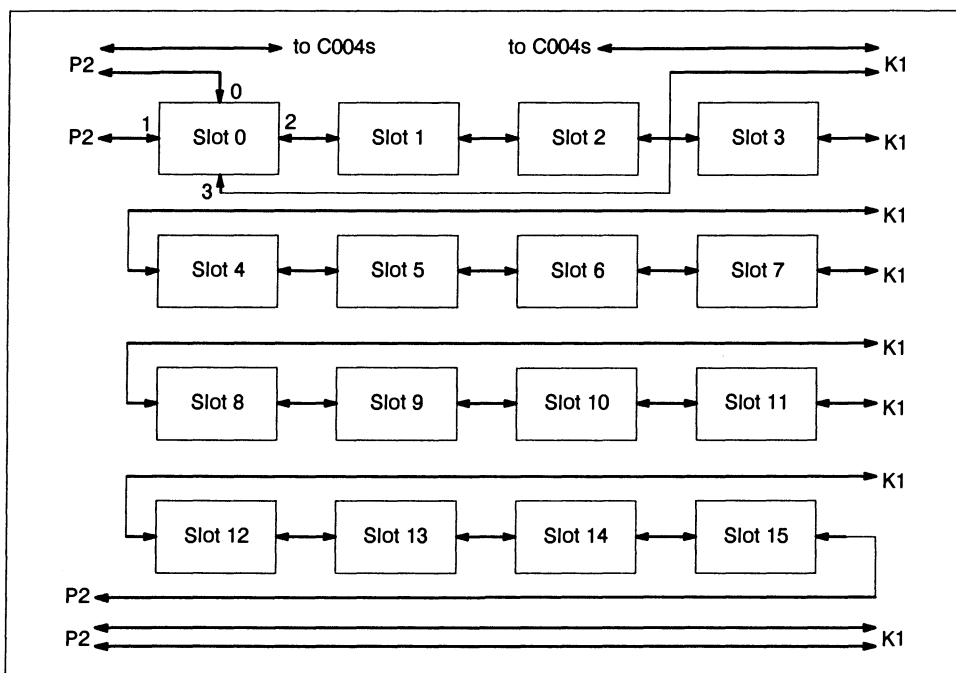


Figure 42.5 Links Available on P2 and K1

P1 Links

Connector P1 has three rows of 32 pins. All the pins in row 'a' are connected to ground. All the pins in row 'b' are link *inputs* and all the pins in row 'c' are link *outputs*. At each of the 32 positions along P1, the three pins from rows a, b and c together carry one link. These signals may be connected to devices with link ports in any way the user desires, as long as the correct electrical precautions required when dealing with links are taken into account. (see 'Link Termination').

A special connector with a small PCB attached and press-fit pins fitted into this PCB is supplied with the IMS B012. This 'mini-backplane', when fitted to P1, allows standard INMOS link cables to be plugged into P1 links. Note that these link cables are not designed for use in arduous physical environments (vibration, corrosion and pulling on the cable). This is why the IMS B012 is provided with standard DIN 41612 connec-

tors. The user may design a connection method for attaching signals to the board which best suits the particular application.

The 32 links available on edge connector P1 are numbered, for reference, starting at 'edge link 0' on pins 1 (a,b and c) through to 'edge link 31' on pins 32 (a,b and c) (see figure 42.6). This numbering scheme is for convenience and there is no obvious mapping between these numbers (the order on the edge connector) and the links to which they are connected on the IMS C004s.

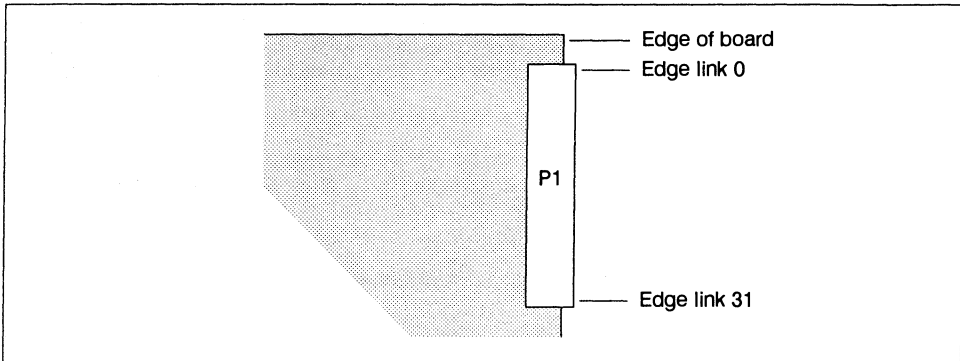


Figure 42.6 P1 Connections

As explained in 'Link Connections', the IMS B012 link switching organisation, using the two IMS C004s, does not allow complete freedom to connect any link to any other link. The following table shows which P1 edge connector links (numbered as above) may be connected to which links on the slots (via the IMS C004 link switches):

P1 Edge Link	To TRAM slot links	P1 Edge Link	To TRAM slot links
0	3	16	3
1	3	17	0
2	0	18	0
3	0	19	3
4	0	20	0
5	0	21	0
6	3	22	3
7	3	23	3
8	3	24	3
9	3	25	3
10	0	26	3
11	0	27	3
12	3	28	0
13	0	29	0
14	0	30	0
15	3	31	0

Table 42.1

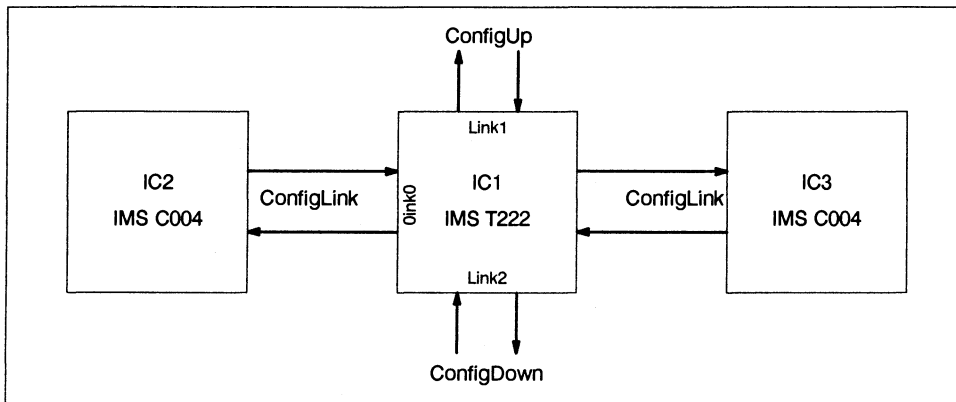
The link connections on connector P1 are intended mainly for communication between the IMS B012 and other boards. However, it is also possible to use these P1 links and the IMS C004 link switches to switch

link connections for an external system. For instance, two IMS B012 boards in a card cage, unpopulated with TRAMs, may act as a 'programmable backplane' to other boards in the card cage. The connections between these boards and the IMS B012s being hard-wired.

Switch Configuration Transputer

The IMS C004 devices are controlled by an IMS T222 16-bit transputer. The IMS T222 has four links. Links 0 and 3 are connected to the two IMS C004s (link 0 to IC2 and link 3 to IC3). Link 1 is available on edge connector P2 and is called *ConfigUp*. Link 2 is also available on P2 and is called *ConfigDown*. The organisation of these links is shown in figure 42.7.

Configuration data for the IMS C004 is fed into one of the IMS T222's links (*ConfigUp*) from the master configuration system which must be connected to P2. The configuration system could be one of the TRAMs on the IMS B012, provided that one of its links may be connected to *ConfigUp*.



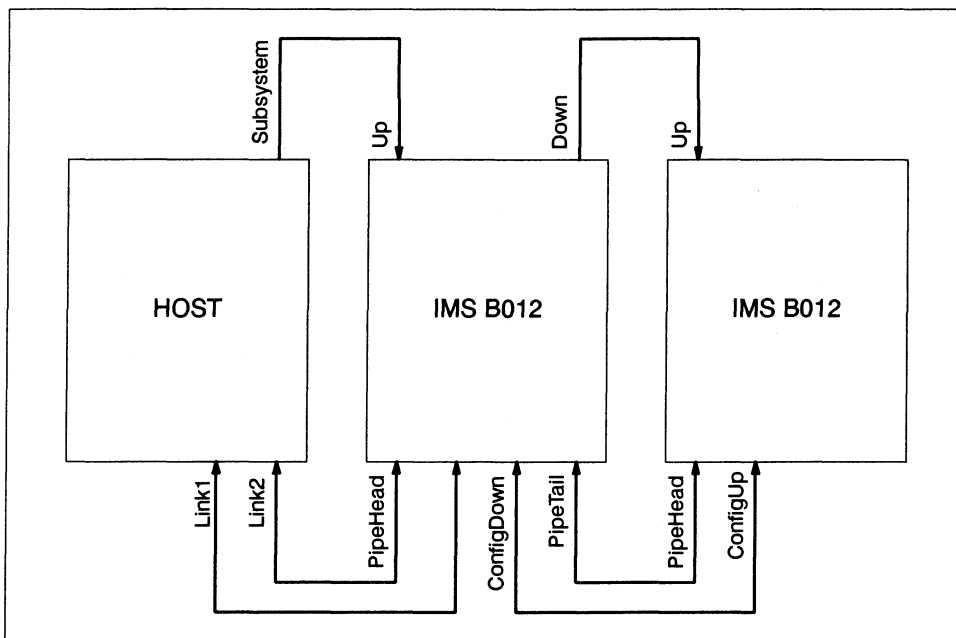


Figure 42.8 Multiple-Board Daisy-Chain

Reset, Analyse and Error

The Reset, Analyse and Error pins of TRAMs (and transputers) will be referred to collectively as 'system services' in this section. The system service signals are used to reset TRAMs and transputers, to place transputers in an analyse state (for debugging) and to carry the fact that an error has occurred in one processor in an array back to some host system which will deal with the error condition.

Some TRAMs and most evaluation boards are capable of generating the system services for other TRAMs and transputers. This is called a 'subsystem' control capability. The IMS B012 can be connected to another board with subsystem control and can also accommodate one TRAM with subsystem control. Furthermore, the IMS B012 can generate subsystem control signals for other boards.

System services for TRAMs are slightly different to system services for boards since the Reset and Analyse signals are active low for boards and active high for TRAMs.

The TRAMs and other circuitry on the IMS B012 splits into two sections for system services. System services for slot 0 come from the 'Up' pins on edge connector P2. System services for slots 1 to 15 and IC1 (the IMS T222) can either come from 'Up' as slot 0, or from the Subsystem pins of slot 0, depending upon the state of switch 6.

The IMS T222 Error pin is unconnected so an error condition on IC1 can not propagate into the TRAM array.

Note that slot 0 is the only slot which has subsystem pins and that in order to use these pins it is necessary to have a module with subsystem capability installed in slot 0.

The system service signals for slot 0 are buffered and output on edge connector P2 as the 'Down' pins. This allows system services for multiple boards to be daisy-chained, the 'Down' of one board being connected to the 'Up' of the next.

Figure 42.9 shows the complete organisation of the system services (reset, analyse and error) on the board.

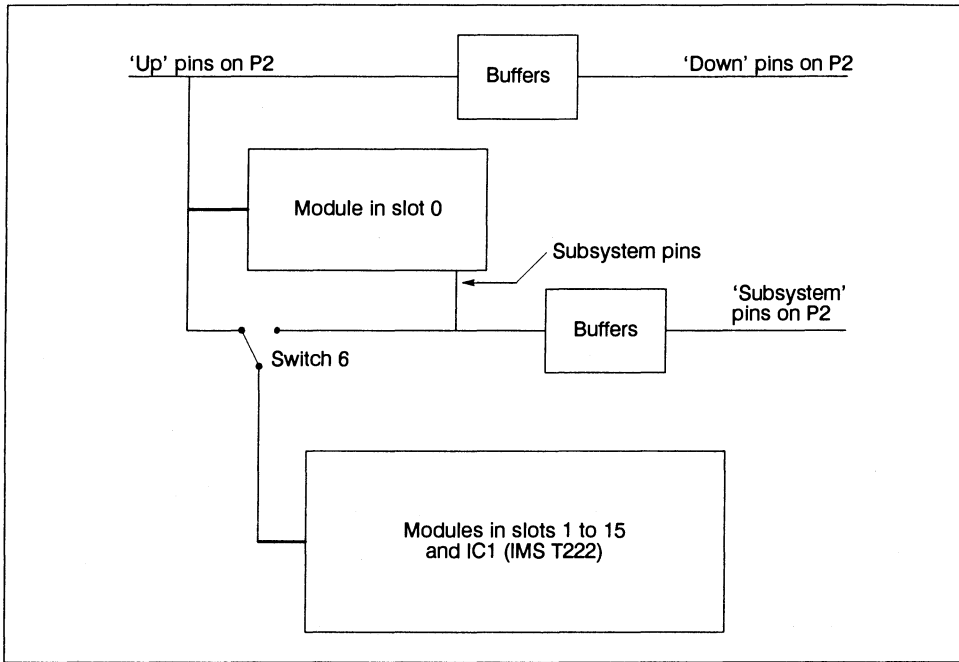


Figure 42.9 IMS B012 System Services Organisation

Reset and Analyse signals presented to the IMS B012 on connector P2 should have minimum low pulse widths of 1 millisecond. The subsystem pins of slot 0 are also buffered and are available on edge connector P2 as the 'Subsystem' pins.

The two IMS C004 link switches have a reset pin that is driven by a power-on-reset circuit. The IMS C004s can be also soft-reset by a command from the IMS T222.

The IMS T222 has 4 Kbytes of on-chip RAM. It also has an external memory interface. Circuitry on the IMS B012 is connected to the IMS T222's external memory interface which allows the reset signal to the IMS C004s to be controlled from the IMS T222. By writing a *one* into bit position *zero* in any external memory word, the reset signal to the IMS C004s is *asserted*. Similarly, by writing a *zero* into bit position *zero* in any external memory word, the reset signal to the IMS C004s is *de-asserted*.

Note that if the IMS T222 (IC1) writes to external memory and sets the IMS C004 reset signal, subsequent resetting of the IMS T222 will not alter the level of the IMS C004 reset signal. Note also that if the IMS T222 reads from any location in its external memory space then the IMS C004 reset signal will be set to an unpredictable level.

Link Termination

INMOS serial links have two signals, linkIn and linkOut. If a link is to be connected over a distance or between boards then some extra discrete components are needed. The linkOUT signal must be series terminated to match its load, and the linkIn should have a diode to VCC for ESD protection and a pulldown resistor to prevent the receiving transputer booting itself from a floating linkIn (see figure 42.10). The whole question of link connections is covered in detail in INMOS *Technical Note 18*.

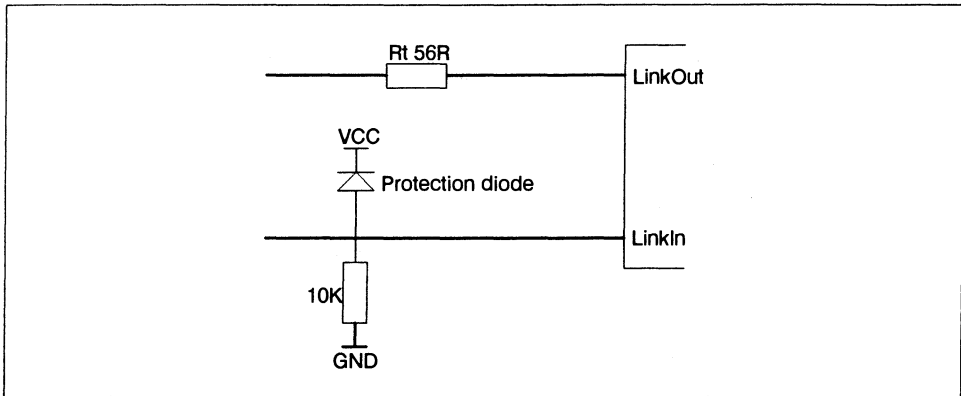


Figure 42.10 Link Termination and Protection

- 1 All links on TRAMs have the link termination and protection components on the module. The Pipe-Head and PipeTail link connections from P2 are directly connected to the module pins (and are therefore terminated). Link 0 from slot 0 is also directly connected to P2.
- 2 The ConfigUp and ConfigDown link connections to the IMS T222 are connected to P2 via termination and protection components.
- 3 All the links on P1 (from the IMS C004s) are terminated and protected.
- 4 The link signals from the IMS C004s which would usually be connected to link 0 of slot 0 (via the jumper connector on P2) are terminated and protected.
- 5 The link signals from the IMS C004s which would usually be connected to slot 0, link 3 (via K1) are terminated and protected since it is possible to route these signals directly to an edge connector (P2).

This means that any link available on the edge connector of an IMS B012 is correctly terminated and protected.

Error Lights

Three yellow LED indicators are mounted on the edge of the board, opposite P1 (see figure 42.11). An indicator will be lit when a module asserts its error pin. One LED, LD1, monitors error from slot 0. The other two LEDs, LD2 and LD3, monitor error from the modules on the front row (not including slot 0), and back row of slots respectively. The front row is the group of seven slots situated along the front-panel side of the board (not including slot 0). The back row is the group of eight slots situated along the opposite edge of the board (see figure 42.11).

When the IMS B012 is installed in a card cage, LD1 is the lower of the LEDs; LD2 is the middle one and LD3 is the upper LED.

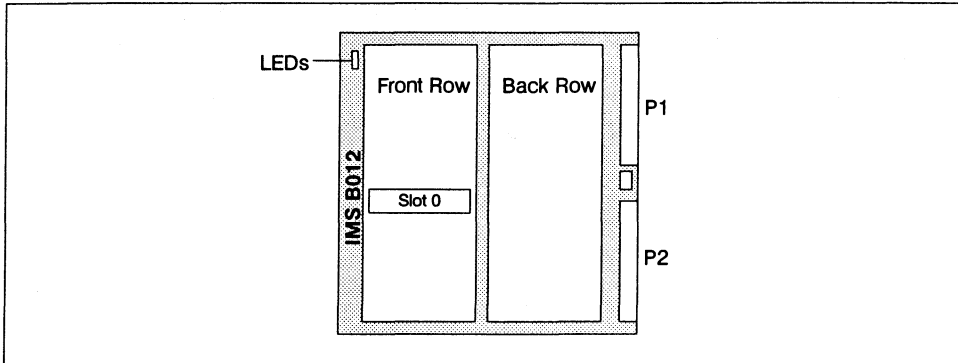


Figure 42.11

User Power Connector

A four pin power connector (designation P3) is mounted near the front edge of the board as shown in figure 42.12. P3 is wired to 0v, +5V and via a wide PCB track to 2 pins on P2. This connector type is the kind used on most floppy-disk drives and when the appropriate pins on P2 are wired to +12V, P3 may be used to power disk drives or similar equipment. Users may take other power signals to P3, such as ECL power supplies.

Pin 4 is connected to connected to 5V, pins 3 and 2 are connected to 0v, pin 1 is pins 3a and 3c on P2. Pin 4 is the top pin when the board is viewed as in figure 42.12.

These power pins can carry up to 3A of current and pin 1 can have up to 50V with respect to GND.

There is a pin post fitted in one corner of the board (marked GND on the silk screen). This is connected directly to the 0V plane and can be useful for attaching 'scope probe ground leads.

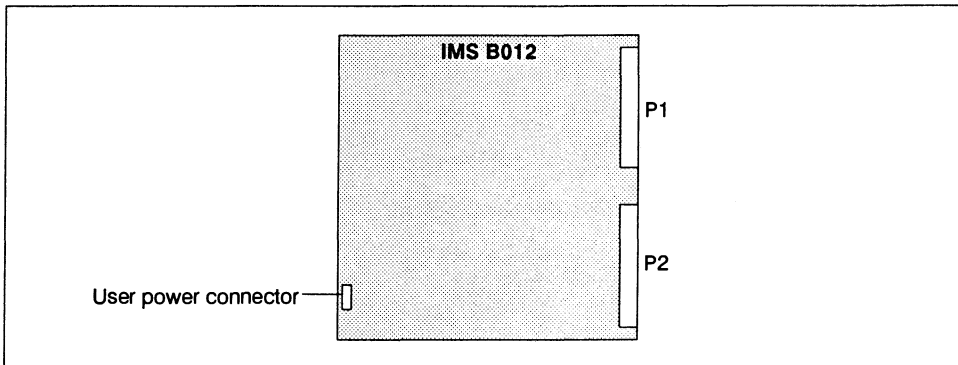


Figure 42.12 P3 Position

Uncommitted Pins

Nine pins on connector P2 are brought to a row of pads near to the connector at the edge of the board (see figures 42.13 and 42.14). These pads, designated P4, may be hard-wired into any circuitry on the B012 by the user for special applications. Possible uses would be RS-232 serial lines, analog signals and extra subsystem control signals. The remaining two pads of P4 are connected to ground. Pin posts can be inserted into the holes which make up P4. The posts could take a single-in-line connector similar to those used in the IMS B012 cable set.

These pins should carry no more than 50mA of current at no more than 25V with respect to GND. Note that these signals, although they are short, have not been designed to carry analog signals and may be susceptible to crosstalk.

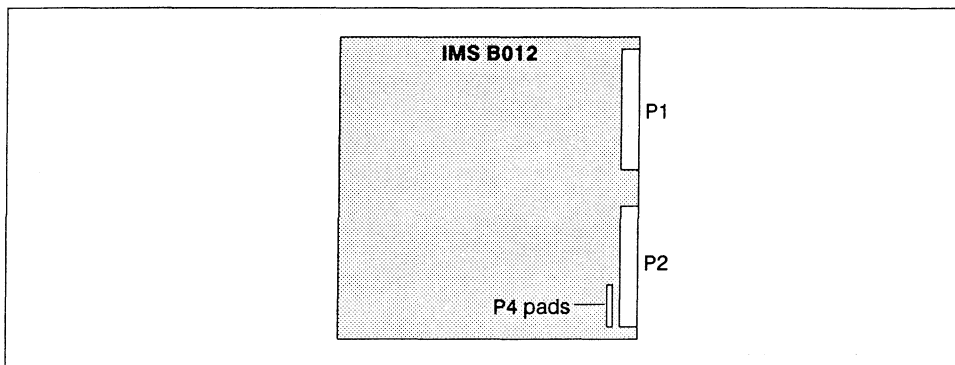


Figure 42.13

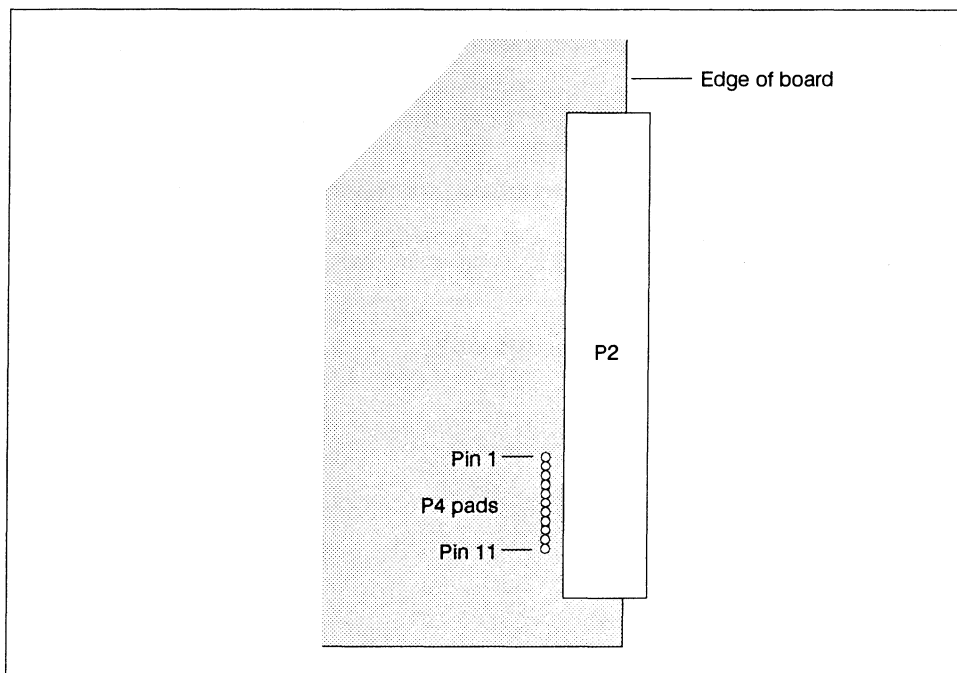


Figure 42.14

42.2 Specifications

Mechanical Details

The IMS B012 is designed to accord with DIN 41494 and IEC 297 standards. The board dimensions are 220mm x 233.5mm with a nominal board thickness of 1.6mm. The supplied front panel width is 4HP (approx 20mm). This is compatible with a board-to-board pitch in a card cage of 0.8in. M2.5 fastening bolts are provided on the front panel, these mate with tapped holes in the card cage and fix the board securely. Front panel handles allow the board to be removed from the card cage. The front panel is *required* when operating the IMS B012 in a card cage, both for mechanical rigidity and to give correct cooling air flow.

Thermal Information

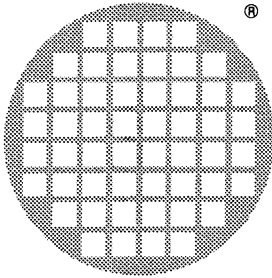
Adequate air flow must be provided to maintain the components on the board within their operating temperature. Air flow should run parallel to the board surface and parallel to the front panel. The amount of heat dissipated by the board depends upon the TRAMs fitted. With no modules the IMS B012 dissipates no more than 3W. With modules fitted the maximum dissipation is 67.5W. The cooling air flow for a particular application will probably need to be determined empirically.

A single board operating in static air at room temperature (and not in a rack) will usually not need forced air cooling. This kind of set-up should only be used for lab work and development work. High reliability should not be expected from boards not provided with adequate cooling.

42.3 Ordering Information

Description	Order Number
IMS B012 Double Eurocard Motherboard	IMS B012-1

Table 42.2 Ordering information

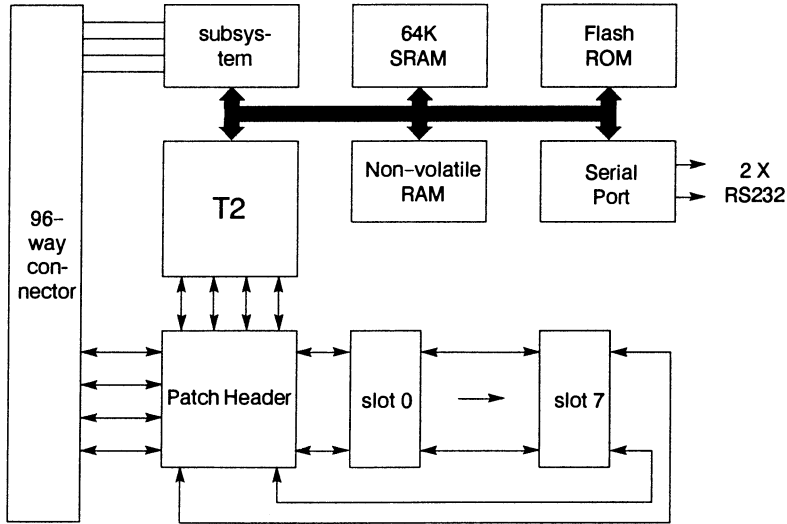


inmos®

IMS B018

TRAM motherboard

Advance Information



FEATURES

- 8 TRAM slots
- IMS T222 transputer
- 64 Kbytes SRAM
- 256 Kbytes Flash ROM
- Two RS232 compatible serial ports
- 8K non-volatile SRAM
- Real time clock
- 4 External links
- 4 Subsystem ports
- 6U VME board profile

GENERAL DESCRIPTION

The IMS B018 is designed to be used in stand-alone applications which do not require a direct connection to a host computer. The board is fitted with an IMS T222 16 bit transputer, controlling all the peripheral circuits.

Flash ROM is provided to allow the TRAM network to be bootstrapped when power is applied. The Flash ROM can also be used for data storage. A battery-backed SRAM is provided for storing small amounts of frequently changed data. The battery also maintains a real time clock when power is removed from the board. 64K of zero wait state SRAM is provided for running programs.

Two serial ports allow connection to a wide variety of computer equipment. Four subsystem ports permit independent control of up to four transputer systems for use in fault tolerant or multi-user systems. Interrupts can be generated by most of the external events that the IMS T222 is required to service. Facilities are provided to reset and reboot the board if the transputer error flag becomes set or if the watchdog timer is not regularly reset. Front panel LEDs can be used to display system status.

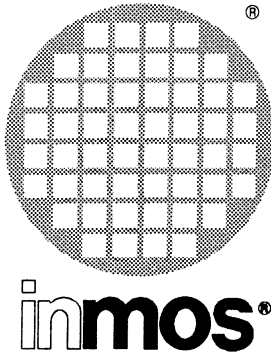
SGS-THOMSON
MICROELECTRONICS

INMOS is a member of the SGS-THOMSON Microelectronics group

Ordering Information

Description	Order Number
IMS B018 TRAM motherboard	IMS B018-1

Table 43.1 Ordering Information



IMS B300

Ethernet connection system

Advance Information

FEATURES

- Compact desktop design, suitable for office or computer room environments.
- Provides full interface to four independent transputer networks over Ethernet.
- Uses TCP/IP protocol suite.
- Choice of single ended or differential link connections.
- Independent diagnostics and test port simplifies installation of software enhancements.
- Activity indicators for each connection.

GENERAL DESCRIPTION

The IMS B300 is a self-contained cased unit providing four transputer links and associated system services for connection to nearby transputer networks housed in their own boxes. The four links are accessed over the IEEE802.3 coaxial network by host computers running the IMS S507, IMS S607 or IMS S707 software products in conjunction with the TCP/IP protocol suite.

The unit supports both the development of transputer programs in a network environment and applications having a transputer-based compute engine used as a network resource.

44.1 IMS B300 Network connection system engineering data

44.1.1 Interfaces

Transputer links and system services signals are driven differentially by the IMS B300 for noise and isolation reasons. This provides a reliable solution for short connections to target systems in the presence of the noise levels likely to be encountered in offices. Translation is required to single-ended signals inside the target system. This can be achieved either by the IMS B415 differential link TRAM or by a similar buffering board which can be mounted in the target equipment.

The connections available on the IMS B300 also support single-ended link and services signals transparently. Single-ended operation is not recommended for reliable operation but will allow diagnostics and prototyping lab-type use to be supported with direct connections to other INMOS boards such as the IMS B008.

The network connection provides an IEEE-802.3 AUI connection. This allows connections to either 10BASE-5, 10BASE-2 and 10BASE-T physical media via suitable "tap boxes". Note that the IMS B300 is not supplied with tap boxes.

44.1.2 Diagnostic Interfaces

Diagnosis and monitoring of a "live" unit is achieved via a serial port on the IMS B300. This expects to communicate with an ANSI compliant video terminal. A set of LED's on the box front panel gives diagnostic information about the state of each interface.

One of the link connections has an associated services "Up" port which can be connected to a B008 or similar board in a PC. This allows field-service diagnosis of a dead unit which can not boot up from its ROMs.

44.1.3 Protocols

The IMS B300 incorporates the IMS F005 firmware, implementing the following protocol elements:

- *BSD 'Socket Library' server* This element extends support for the BSD Socket interface to transputer applications which use the separately supplied compiler libraries.
- *Linkops Connection Server* Provides access to named transputer subsystems over TCP/IP to remote users.
- *TCP and UDP transport services* The TCP implementation provides a reliable connection oriented transport layer, designed to meet Internet standards RFC793 and RFC1122. UDP provides a datagram service, implemented to RFC768 and RFC1122. Both of these protocols are available to transputer applications via the Socket Library interface.
- *IP Network Layer* This layer is designed to meet Internet standards RFC791 and RFC1122 for IP layer behaviour.
- *Address Resolution Protocol* The dynamic Ethernet address resolution protocol, designed to meet Internet standards RFC826 and RFC1122.
- *Ethernet Data Link Layer* The Ethernet interface in the device encapsulates IP layer datagrams within Ethernet packets. The packetisation used is as specified in Internet Standard RFC894 (*i.e.* Ethernet V2 Packetisation) The Data Link Layer requirements of RFC1122 are also met by this layer.

44.1.4 Performance

A raw TCP data-rate of several hundred kilobytes/second is supported. This is measured as the average transfer rate achieved sending TCP data from a Socket Library application on one of the IMS B300 subsystems to a similar socket library application on a fast UNIX host, over a local Ethernet. Some reduction in total data bandwidth may occur if used with a host server or when multiple IMS B300 links are operating concurrently.

44.1.5 Specification

Feature		Unit
Power dissipation	25	W
Operating temperature	10-40	°C

Table 44.1 IMS B300 specification

Links and system services signals are buffered to levels compatible with EIA RS-422. Inputs will also receive TTL levels and one side of the differential outputs can be used to drive a TTL-compatible input. The serial interface provides a functional subset of an EIA RS-232 DCE.

The IMS B300 is powered from AC mains and is compatible with worldwide mains supplies.

The unit is shock and vibration resistant to a level consistent with office use and air-freight.

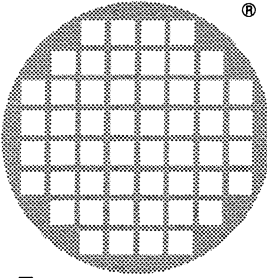
44.1.6 Ordering Information

Description	Order Number
IMS B300 network connection device	IMS B300-1

Table 44.2 Ordering information



Associated Hardware Products

**inmos**®

IMS B250

VME Rack

Engineering Data

FEATURES

- 12 slots for 6U VME boards
- Blanking panels provided for unused slots
- Built-in power supply capable of delivering 40A at 5V and 2 X 6A at 12V
- Built-in forced air cooling
- Accommodates INMOS VMEbus boards such as the IMS B014 and IMS B016
- Disk storage can be fitted to VME slots
- Can be configured to meet FCC regulations
- 110-120V or 220-240V operation

GENERAL DESCRIPTION

The IMS B250 VME rack is a cabinet that will accommodate up to 12 INMOS VME boards, such as the IMS B014 VMEbus motherboard and IMS B016 VMEbus master/slave board. The IMS B250 will also accept other VMEbus boards including disk storage.

The B250 provides a simple means of connecting transputer boards together with the necessary power supply and cooling requirements to provide the potential for supercomputing power.

45.1 Front panel

The VME rack is provided with the following front panel indicators:

Mains Power

+5V

+12V

-12V

System Fail (monitors VMEbus SYSFAIL*)

A reset button is also provided on the front panel (asserts VMEbus SYSRESET*).

45.2 Specification

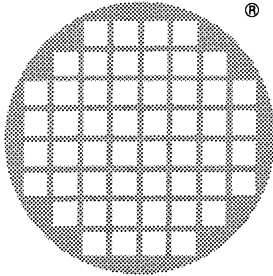
		Units
External dimensions		
Height	360	mm
Width	530	mm
Depth	420	mm
Power supply (forced air ratings)		
5V	40	A
24V	6	A
12-15V (1)	6	A
12-15V (2)	6	A
5-15V	3	A
Maximum output power (total)	300	W

Table 45.1 IMS B250 specification

45.3 Ordering Information

Description	Order Number
VME Rack 240 Volt operation	IMS B250-1UK
VME Rack 120 Volt operation	IMS B250-1US

Table 45.2 Ordering Information



inmos®

IMS CA12

Card Frame Adapter

Engineering Data

FEATURES

- 6U to 9U card frame adapter for SUN workstations
- Enables 6U size VME boards (for example IMS B014 and IMS B016) to be used with SUN workstations with 9U size backplanes
- Isolates the P2 user-defined pins from the backplane

GENERAL DESCRIPTION

Some VMEbus compatible card cages, notably SUN workstations, make use of the user defined pins on connector P2. It is extremely important that INMOS VME cards such as the IMS B014 and IMS B016 are not plugged into such a card cage because permanent damage to the VME card and/or SUN can result.

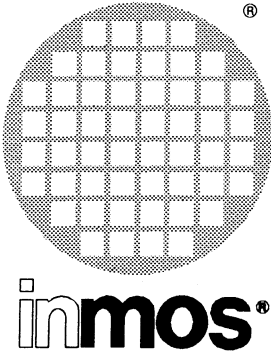
Although this restriction only applies to some slots in some kinds of SUN (and probably other card-cages) users should always be aware of this risk.

The solution required for the SUN is to use the IMS CA12 card frame adapter which isolates the P2 user-defined pins from the backplane.

Ordering Information

Description	Order Number
6U to 9U VME card frame adapter for SUN	IMS CA12

Table 46.1 Ordering Information



Cables for Board Products

The following cable sets are available to complement the INMOS range of board products. Sufficient cables are included with each of the INMOS board products to build the most common configurations. However, where more sophisticated systems are required, it will sometimes be necessary to use additional cables. The table below indicates the number of each cable type included in each of the available cable sets.

Part Number	Cable Reference Code														
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
IMS CA01					10	2	1	1							
IMS CA03	10	10	1	1											
IMS CA09									3	10	10				
IMS CA10												1			
IMS CA11													1		
IMS CA13														1	
IMS CA14															1

Table 47.1 Cable sets

Code Description

- A 10cm long link cable, terminated by two standard link connectors
- B 50cm long link cable, terminated by two standard link connectors
- C 1m long link cable, terminated by two standard link connectors
- D 2m long link cable, terminated by two standard link connectors
- E 10cm long reset cable, terminated by two standard reset connectors
- F 50cm long reset cable, terminated by two standard reset connectors
- G 1m long reset cable, terminated by two standard reset connectors
- H 2m long reset cable, terminated by two standard reset connectors
- I 3 x 1 pin strip, suitable for routing subsystem signals between TRAMs and motherboards
- J 8 x 1 TRAM pin extender strips, suitable for raising TRAMs above components mounted on motherboards

- K 8 x 1 pipe jumper strips, suitable for maintaining the link pipeline structure on motherboards
- L Pixel bus terminator module, suitable for use with the distributed graphics system (IMS B409-1)
- M 37-way 'D' type connector to standard link pin convertor, suitable for use with some motherboards (for example IMS B008-1, IMS B014-1, IMS B017-1)
- N 1m long RGB cables terminated at one end with SMB connectors and at the other with BNC connectors. These are suitable for use with the graphics interface TRAMs (for example IMS B409-1, IMS B419-4)
- O 50 cm long ribbon cable designed to connect a GPIB interface TRAM (for example IMS B421-10) to the standard IEEE-488 bus.



Application Notes



Dual-In-Line Transputer Modules (TRAMs)

(INMOS Technical Note 29)



48.1 Background

In the early days of the transputer, INMOS built a number of transputer evaluation boards. Most are the same size (220mm x 233.4mm), have different transputer configurations, different amounts of memory, transputer graphics or several transputers.

INMOS has also produced boards to fit some particular computers.

The need

It would have been nice if we had been able to offer all the different transputer configurations to fit into these and other personal computers. But instead of about ten different designs of boards, this would have meant 30 different designs. And there was market demand for transputers to plug into VME, to VAX, to SUN, to other workstations, process control computers, minicomputers, mainframes. And there was further demand for more configurations, such as more memory per transputer, more transputers with less memory, or the same memory in much less space, graphics and other different peripherals.....

Clearly to produce all these different transputer configurations, to plug into all these different computers, would need over 100 different board designs. Even if INMOS could design those, it would be foolish to stock and sell so many different designs. But a genuine market demand existed to be met. Somehow we had to separate the transputer configuration from the computer and its size and shape of board.

Meeting the need

A small range of transputer modules (TRAMs), implemented as modular subsystems, and a small range of motherboards with sockets for the TRAMs, offered this separation.

Users can mix and match different physical sizes of TRAMs – TRAMs with different memory sizes and with different functions. By mixing and matching, many more than 100 different combinations are possible.

An advantage to many customers who have the expertise in interfacing to their own computers is that they can design their own motherboards, and use the ready-built transputer configuration supplied as TRAMs. This greatly reduces the time needed to prototype a transputer system. Many customers are also finding it easier to continue to build their systems based on TRAMs.

The building block

In effect the TRAM is a board level transputer, with a very simple standardized interface. The building block concept is practically realized by integrating memory and peripheral functions on board, and by limiting the pin out to 16 pins (although some modules use several sets of these 16 pins). It is just as easy to build transputer circuits with modules as it used to be to build logic circuits out of TTL.

Several of the TRAMs are densely packed, offering thousands of MIPs, hundreds of MFLOPs and many megabytes, all on a few motherboards in a small box.

Why so small?

The size comes from considering how small a transputer could become. As the chip is about 1cm square, it would not fit with a 0.3" 16 pin DIP, but it would fit into a 0.6" 16 pin DIP. Put four of these on a regular prototyping board with rows of sockets on 0.3" centres and you have a set of pins 9–16 just 3.3" away from pins 1–8. Add enough at each end for mechanical fixing and width for a PGA to give the final size.

So the size was primarily chosen to fit standard prototyping boards. Conveniently, the size also fits the IBM PC, VME boards, as well as a host of other computers.

48.2 Introduction

TRAMs are small subassemblies of transputers (or other components with INMOS links), a few discrete components, and sometimes some RAM and/or application specific circuitry. They:

- interface to each other via INMOS links

- have a standard pinout

- come in a range of standard sizes

The basic size of a TRAM is 1.05" by 3.66" overall, about half the size of a credit card. This basic size is referred to as Size1. Larger TRAMs can be up to 8.75" by 3.66", which fits comfortably on an IBM PC board or on a VME board (this largest size is referred to as Size8). Smaller TRAMs (hybrids or silicon, not yet implemented) can be as small as a 16 pin DIP with leads on 0.6" centres.

The standard pinout and standard sizes of TRAMs make it very simple for users to build customized motherboards with sockets for TRAMs. These can either be in prototype form (Perfboard, Vectorboard or Vero-board), or in printed circuit form.

TRAMs may be plugged into the TRAM sockets on any of a wide range of TRAM motherboards made by INMOS and many of INMOS' customers, to fit most of the popular computers and busses.

Most of the motherboards include C004 link switches, so that users can configure their own networks to match the structure of the problem they are trying to solve.

The TRAM standards referred to above are independent of:

- transputer type (IMS T222, T414, T800, T425, T801, etc.)

- number of transputers (1, 4, 8, 12, 16 are all possible)

- wordlength of transputer (16 bits on T222, 32 bits on T800)

- speed (from 17.5 to 30MHz and beyond)

- function (transputer plus RAM, disk control, other peripheral control)

- memory size (no external RAM up to many megabytes)

- package (68, 84, 100 pins. PGA, or surface mount PLCC and PQFP)

- implementation (through-hole PCB, surface mount PCB, hybrid, silicon)

48.3 Functional description

48.3.1 Pinout of size1 module

The pins include four INMOS links, which require no off-module buffering.

Table 1 shows the pinout. This pinout has been chosen partly to simplify layout of the motherboard, and partly to simplify the layout of the TRAM.

1	Link2out	Link3in	16
2	Link2in	Link3out	15
3	VCC	GND	14
4	Link1out	Link0in	13
5	Link1in	Link0out	12
6	LinkSpeedA	notError	11
7	LinkSpeedB	Reset	10
8	Clockin(5MHz)	Analyse	9

Table 48.1 Standard TRAM pinout

When LinkSpeedA and LinkSpeedB are both low, the TRAM links operate at 10Mbits/s. When they are both high, the links operate at 20Mbits/s. Other states of these pins are reserved for future enhancements.

The notError signal is driven by an open collector transistor so the signal can be wire ORred. This allows for the error line to be bussed in the same way as Clock, Reset, and Analyse. The fan-in of the notError signal must be controlled, and it is recommended that no more than ten notError outputs are wired together.

Pin 1 is marked by a silk screened triangle.

48.3.2 Pinout of larger sized modules

Figure 48.1 shows two adjacent Size1 TRAMs side by side. Notice that the orientation of the two modules is different. This difference in orientation serves two purposes: cooling of Size1 modules is improved; and it makes it possible at some future date to have Single-In-Line modules.

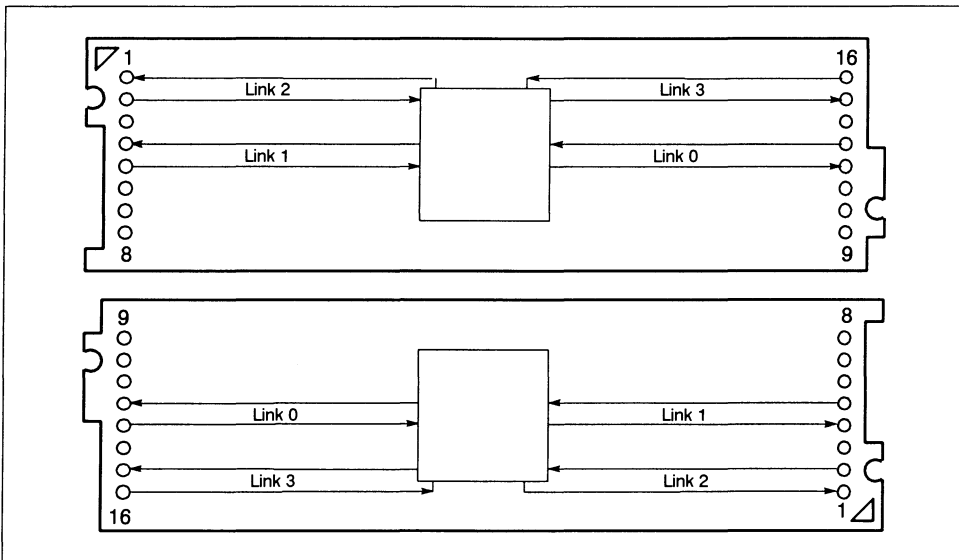


Figure 48.1 Orientation of adjacent Size1 modules

Many modules, and all the early products IMS B401 to B405, contain a single transputer, and so do not need more than one set of 16 pins for electrical signals. Modules larger than Size1, however, are assembled with extra sets of 16 pins; the extra pins give mechanical support, allow modules to be stacked, and provide extra GND and VCC pins. A Size2 module with one transputer is shown in figure 48.2a.

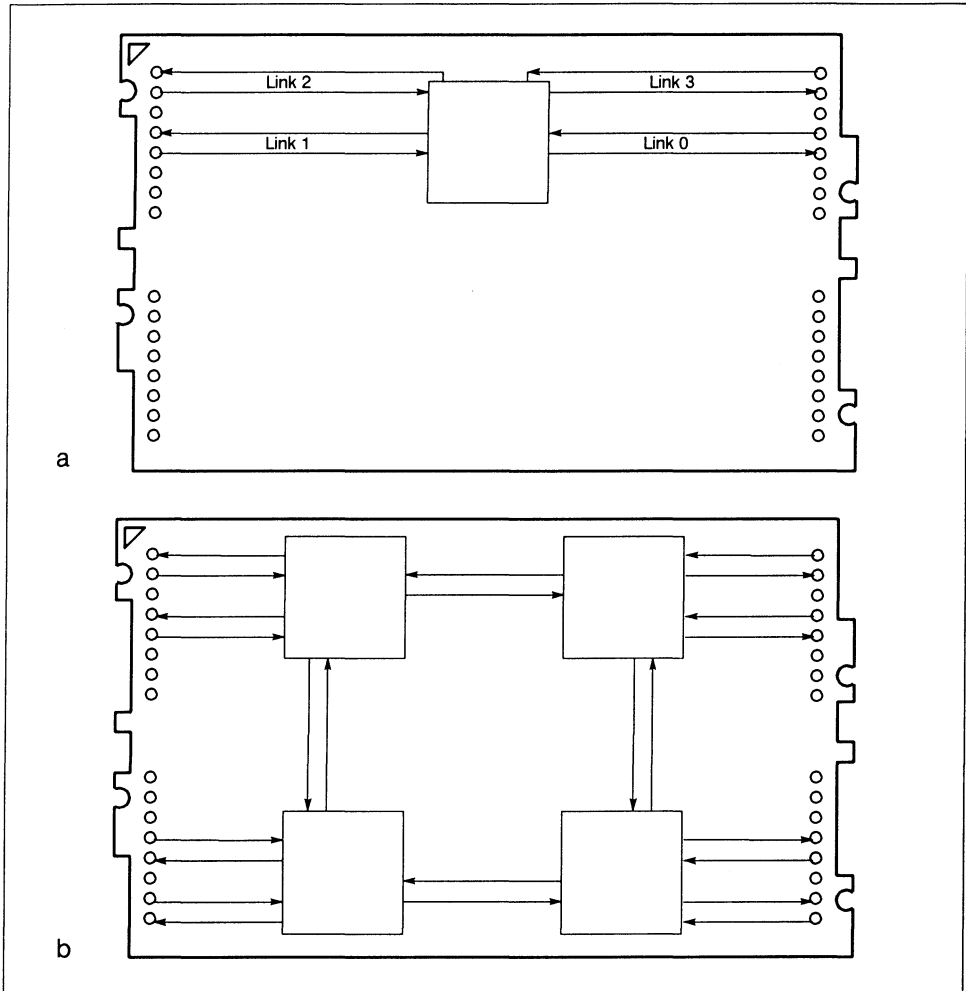


Figure 48.2 Size2 TRAMs with one and four transputers

TRAMs may be built with more than one transputer, or with transputers having more than four links. An example of a possible TRAM with more than one transputer is shown in figure 48.2b. This has four transputers connected as a square, in the same way as the IMS B003 and B006. (In practice, if INMOS were to produce a TRAM with four transputers, the links would probably be routed to make better use of standard motherboard connections.)

The detailed pinouts of larger modules are shown with the mechanical details in section 48.8 and assume that each TRAM has a single transputer, with four links.

Notice that the Size2 module and the Size4 module have the pins which are actually used at one end. The Size8 module (when it has a subsystem capability) has the pins which are used in the middle.

48.3.3 TRAMs with more than one transputer

Standards for pinout of transputers with more than one transputer are to be defined.

48.3.4 Extra pins

TRAMs may include application specific circuitry which requires pins other than the standard 16 pins. Examples are peripheral controllers or pipelines used for graphics or signal processing. The recommended connector for these is a strip of pins on 0.1" grid, such as a stripcable socket will attach to.

48.3.5 Subsystem signals driven from a TRAM

It is useful for TRAMs to be able to control a network of transputers and/or more TRAMs. Such a slave network is known as a *subsystem* of the master, and the set of control signals from the module are described as a *subsystem port*.

The subsystem port consists of three signals: **SubsystemReset** and **SubsystemAnalyse**, which enable the master to reset and analyse its subsystem; and **SubsystemnotError**, which is used to monitor the state of the error flag in the subsystem. The polarity of these signals is such that a motherboard can be built with a master TRAM controlling slave TRAMs via its subsystem port with no buffering or gating. (Note that a change of polarity may be required for a subsystem port which goes off the motherboard.)

The three subsystem signals are located on low profile sockets which are positioned 0.1" inside the standard module pins 1-3. This is illustrated by figure 48.3.

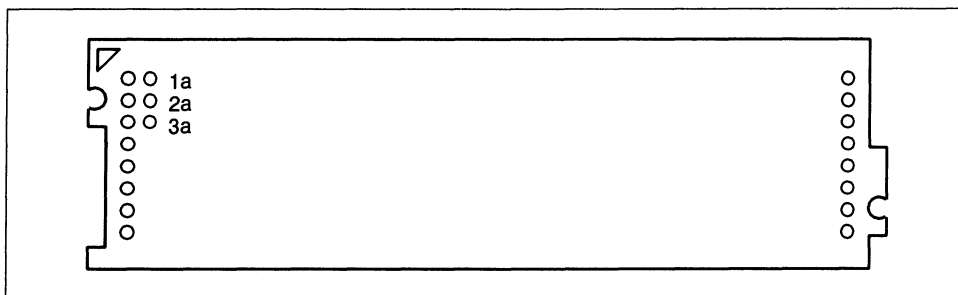


Figure 48.3 Location of subsystem sockets

The pinout is as follows:

Pin	Signal
1a	SubsystemnotError
2a	SubsystemReset
3a	SubsystemAnalyse

The sockets are fitted into the module PCB upside-down. The motherboard into which the module is plugged will also have three such sockets in the corresponding positions, but fitted from the component side in the usual fashion. The connection between the module and the motherboard is then made by a double-ended header, strip (see figure 48.4). This arrangement ensures that if the subsystem port of a module is not used, the module remains mechanically compatible with modules which do not have subsystem ports.

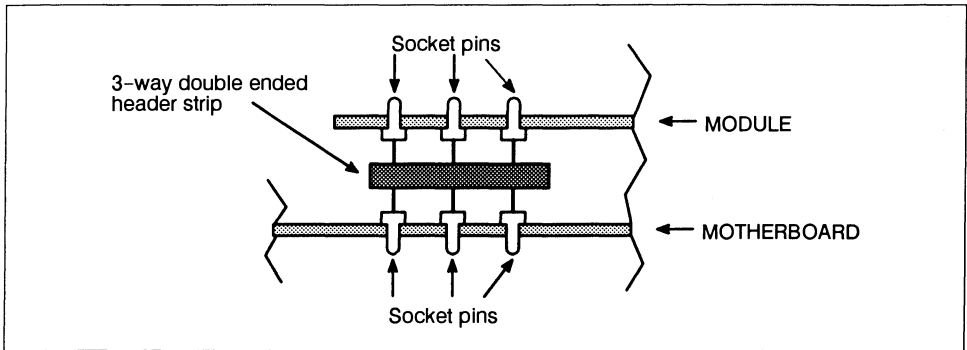


Figure 48.4 Subsystem port connections

Subsystem registers

The subsystem is controlled by reading and writing to addresses in positive address space (i.e. location zero onwards). On all INMOS evaluation boards and TRAMs, two BYTE locations are used, where each byte is the least significant byte of a 32 bit word. A further two locations control parity generation logic, which will be described in section 48.3.6. These four locations are permitted to repeat throughout the whole of the positive address space.

The subsystem registers are located at the following addresses for 32 bit transputers

Register	Hardware byte address
SubSystemResetLatch (write only)	#00000000
SubSystemAnalyseLatch (write only)	#00000004
SubSystemnotError (read only)	#00000000

The subsystem port operates as follows:

Writing a 1 into bit 0 of #80000000 asserts SUBSYSTEM Reset;
 Writing a 0 into bit 0 of #80000000 deasserts SUBSYSTEM Reset.

Writing a 1 into bit 0 of #80000004 asserts SUBSYSTEM Analyse;
 Writing a 0 into bit 0 of #80000004 deasserts SUBSYSTEM Analyse.

A 1 read from bit 0 of #80000000 indicates that SUBSYSTEM Error is TRUE.
 A 0 read from bit 0 of #80000000 indicates that SUBSYSTEM Error is FALSE.

The subsystem is reset or analysed under the control of the transputer on the TRAM, but must also be reset when the TRAM itself is reset. To pass the signals on to the subsystem, the following combinational logic is included:

SubsystemReset = Reset OR SubsystemResetLatch
 SubsystemAnalyse = Analyse OR SubsystemAnalyseLatch
 the latches are initialized at power-on to be inactive.

Note that SubsystemError does NOT propagate to the TRAM's notError pin.

Multiple subsystems

TRAMs may contain more than one subsystem port. They should have their locations separated by 16 bytes.

48.3.6 Memory parity

TRAMs may include parity logic for external RAM. The implementation on TRAMs must ensure that there is no way that corrupt data can reach any other transputer.

One way to achieve this is that if a parity error occurs, the wait signal is held active so the memory cycle does not complete. All data in memory is lost, however, when an error occurs, and the memory cycle is slowed down by the parity check.

Parity checking may be enabled or disabled by writing to a parity control register. If parity is enabled and an error occurs, the error is ORed in to the notError signal from the module. Information on the cause of the error can be found by examining the parity status register.

Reset disables parity checking and deasserts MemWait. When the transputer is analysed, MemWait is deasserted and the contents of the parity status register are preserved.

The parity registers are as follows:

Register	Hardware byte address
Parity control (write only)	#00000008
Parity status (read only)	#00000008

The locations are used as described below:

Writing a 1 into bit 0 of #80000008 enables parity error detection;

Writing a 0 into bit 0 of #80000008 disables parity.

Reading the contents #80000008 returns the status of the parity detection hardware.

Bit	Status
Bit 0	Indicates a parity error has occurred.
Bits 1 & 2	Indicate the BYTE in which the error occurred. (Bit 1 is lsb).
Bits 3..n	Indicate the BANK in which the error occurred. (Bit 3 is lsb).

48.3.7 Memory map

The memory map should be of the form:

```

ROM          top of memory
Peripherals
Subsystems
External RAM
On-chip RAM  bottom of memory

```

In the particular case of TRAMs with 32 bit transputers, the memory map should be as follows:

Byte address	Description	Comment
7FFF FFFF		Bootstrap program requires ROM at top of memory.
7FFF FFFE	Boot from ROM	7FFF FFFE will contain a backward jump to the bootstrap.
	Peripherals	If used
0000 000C		These locations must be decoded as a set of four, even if Parity is not used.
0000 0008	Parity status and control	
0000 0004	SubsystemAnalyseLatch	
0000 0000	SubsystemResetLatch	
8FFF FFFF	RAM	Both internal and external RAM
Memstart	RAM	

Substantial logic can often be saved by not fully decoding the hardware address. An effect of not fully decoding the address is that hardware can appear at multiple addresses.

In particular, if the module does not have a subsystem, the RAM can repeat throughout the address space, including the positive address space (above location 0).

The Subsystem and parity locations can also repeat throughout the positive address space.

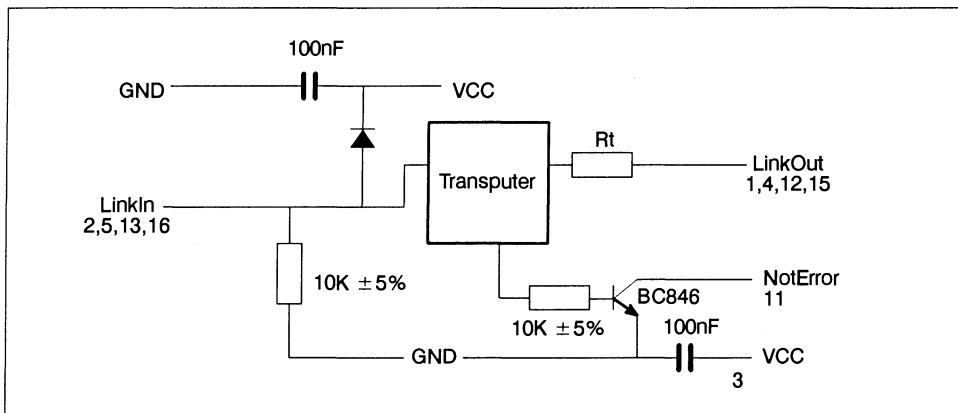


Figure 48.5 Recommended circuit between TRAM pins and transputer

48.4 Electrical description

48.4.1 Link outputs

Link outputs must be terminated so that the combined output impedance of the transputer plus termination resistors is 100 ohms \pm 20%. For the optimum value of resistor, see the appropriate transputer data sheet.

48.4.2 Link inputs

Link inputs may be taken off a module motherboard and so must be protected from positive ESD by a diode to VCC. Signal diodes such as 1N4148 or LL4148 may be used. To prevent an unconnected link input from floating high, link inputs must be pulled down to GND by a resistor, preferred value 10K \pm 5%.

48.4.3 notError output

The notError output is a wired OR signal driven by an open collector or an open drain. Maximum leakage should not exceed 10 microamps. Maximum saturation voltage when the transistor is ON and is sinking 10 mA should not exceed 0.4 V. A suitable transistor is BC846 (SOT23) with a 10K resistor between the transputer's Error pin and the transistor base. The pullup resistor on the module motherboard should draw between 5mA and 10mA when a transistor is ON.

Although the above is conservative and should allow a fan-in of several hundred, it is recommended that the fan-in is limited to 10.

48.4.4 Reset and analyse inputs

These signals are connected directly from the TRAM pins to the transputer. They must always be driven by buffers on the module motherboard. Because the motherboard will often have filters on the Reset and Analyse signals, the Reset pulse width should be much wider than specified for the transputer. Recommended pulse width is 5 ms, with a delay of 5 ms before sending anything down a link.

48.4.5 Clock input

The TRAM must not present excessive capacitance to the clock input signal. The clock input should therefore be limited to a single load, which should be connected to the TRAM pin by a trace no longer than 30mm.

Particular care should be taken on the module motherboard to ensure that the clock input is clean, with fast edges, minimal undershoot, and minimal jitter (see transputer data sheet for clock specification).

48.4.6 notError input to subsystem

The notError input should not have a pullup resistor on the TRAM. The pullup resistor must be on the motherboard.

48.4.7 GND, VCC

Adequate high frequency decoupling capacitors must be used. In particular there should be decoupling capacitors close to the GND pin and to the VCC pin of each TRAM. Recommended value is 100 nF, preferably at least half as many as the module has ICs.

48.5 Mechanical description

In the following, dimensions are quoted in inches for PCB length, width and related dimensions; all other dimensions are quoted in millimetres.

48.5.1 Width and length

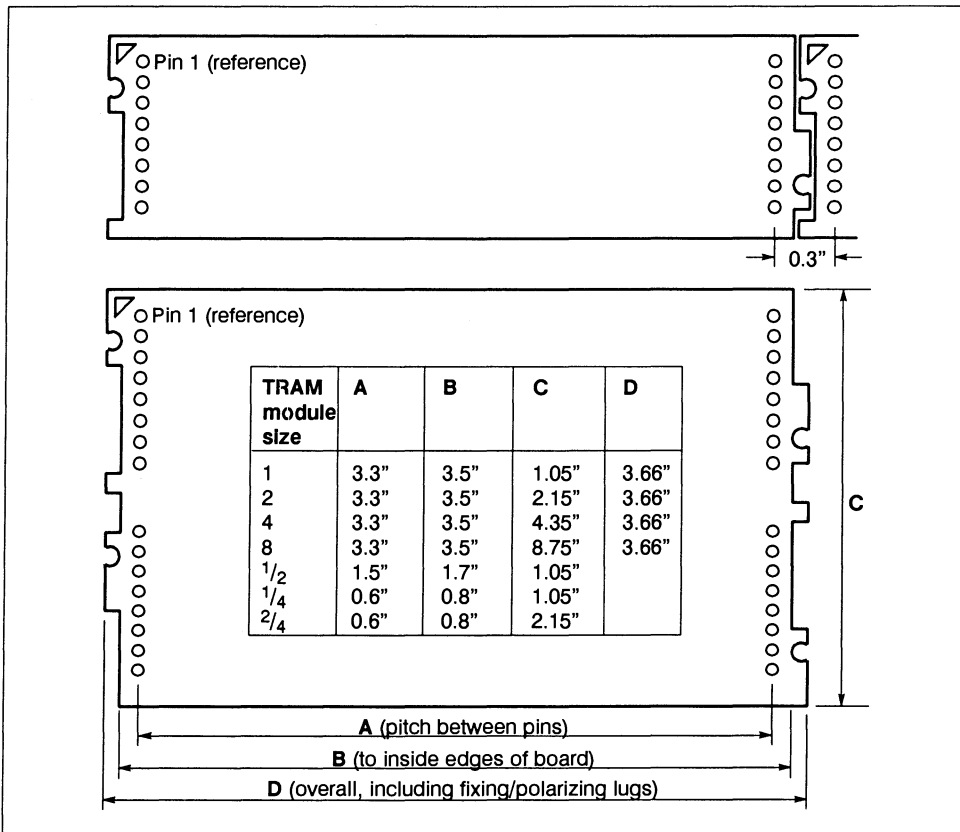


Figure 48.6 TRAM sizes

The basic size of a TRAM is a very wide 16 pin DIP, with 3.3" between the two rows of pins. These TRAMs fit on a 3.6" pitch on their length, and a 1.1" pitch on their width. Extra length is added beyond the pins to hold the pins, to provide for mechanical fixing, and to polarise the module shape.

TRAMs can be made larger than the standard size by keeping the 3.3" between pins and using two or more sets of the 16 pins.

TRAMs can be made smaller than the standard size, down to a 16 pin DIP with 0.6" between the two rows of pins, or 1.5" between the pins. These sizes will normally be used for single chip modules or hybrids.

In general the printed circuit TRAMs are longer than the pitch between the two rows of pins. The TRAMs are also wider than the 0.8" suggested by 16 pins. The small TRAMs may be side-brazed DIPs, as short as 0.8" long. The top drawing in figure 48.6 shows a Size1 module and how the jigsaw pattern fits together between adjacent modules. The lower drawing in figure 48.6 shows the various sizes of TRAM. Detailed dimensions of the different sizes are given in section 48.8.

48.5.2 Vertical dimensions

There are no vertical height constraints for TRAMs. However, keeping the height of a TRAM, both below and above the board, within certain limits allows the TRAM to fit together with other TRAMs and motherboards.

Figure 48.7a shows height specifications, both above and below the TRAM PCB. Figure 48.7b shows how this vertical size fits onto a motherboard which has no components under the TRAM. Figure 48.7c shows the same TRAM fitted above components on a motherboard, using spacer socket strips to gain extra height.

Figure 48.7d shows another height specification which allows components such as zip packaged ICs and SMB connectors to be used on the TRAM, whilst permitting these TRAMs to fit onto motherboards in a 0.8" pitch card cage. Note that this is only possible when there are no components under the TRAM on the motherboard.

The TRAMs are specified to make it possible to stack one TRAM above another in some circumstances. The combination of physical and thermal constraints on stacking has meant, however, that for a number of TRAM implementations stacking is not possible. The figure showing stacked TRAMs (which appeared in earlier versions of this document) has therefore been removed in this version.

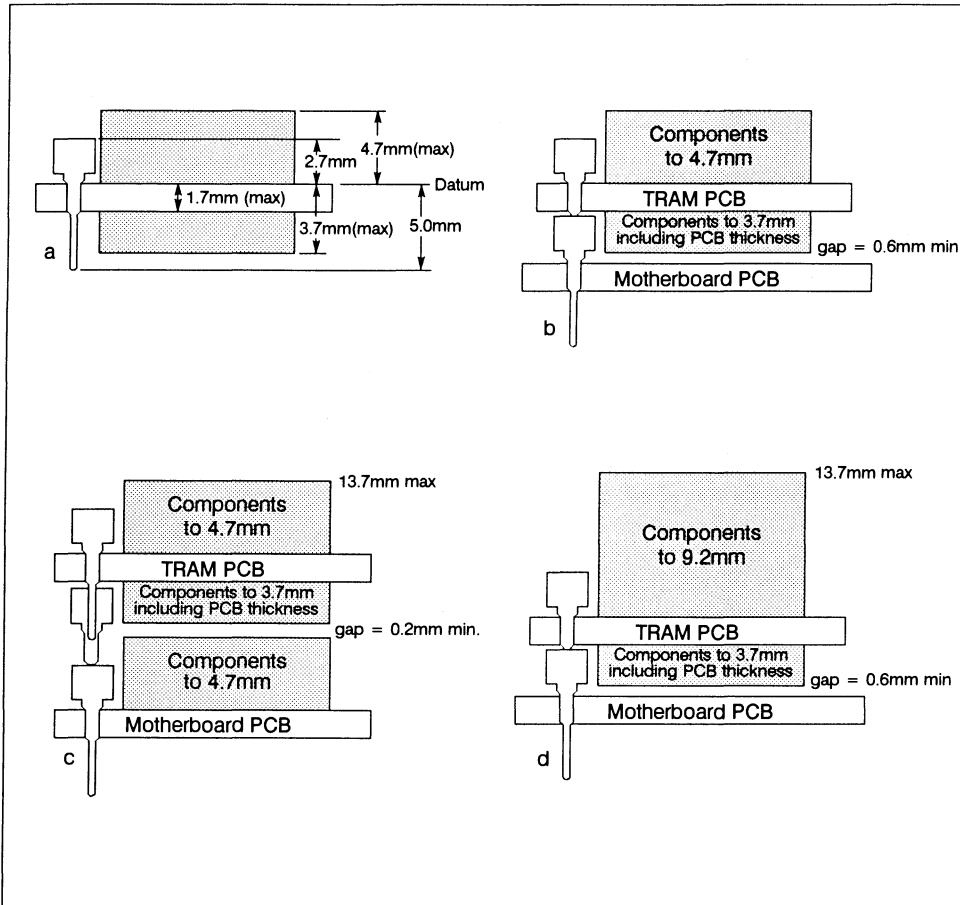


Figure 48.7 Component heights

It is recommended that any component reaching a maximum specified height has an insulating surface.

Note that the datum for component heights on both sides of the TRAM is the component side surface. This datum is also used for the stackable socket to minimize tolerance buildup.

Components must not interfere with the TRAM pins, and so the area shown in figure 48.8 must be left free of components.

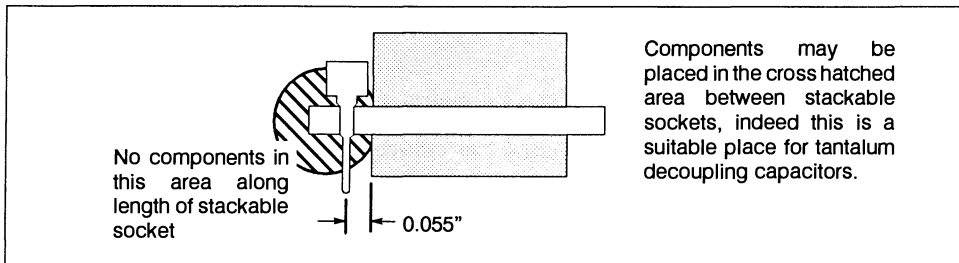


Figure 48.8 Area close to TRAM pins

48.5.3 Direction of cooling

TRAMs should be designed so that cooling air can flow freely across the width of the the module, or in other words parallel to pins 1 to 8 rather than from pin 1 to pin 16. Care should also be taken to ensure that the surface of a module is not too flat: projections cause turbulence which improves cooling.

48.6 TRAM pins and sockets

48.6.1 Stackable socket pin

The stackable pin socket is shown in figure 48.9.

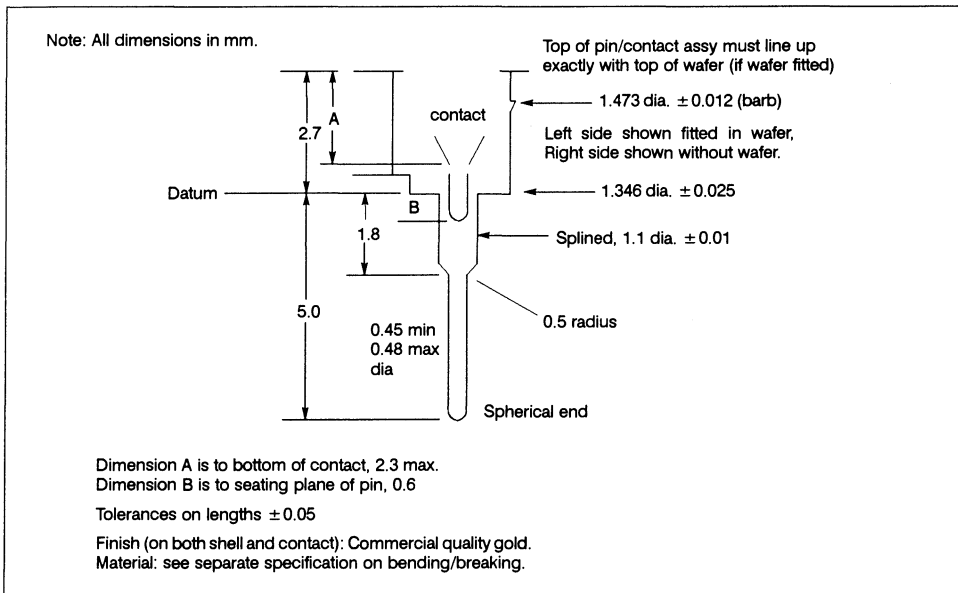


Figure 48.9 Stackable socket pin

Approved manufacturers of the stackable socket pin are (with part numbers): ¹

	Individual socket pin	Strip of 8 sockets
Scott	128-446	15108-128-446

The individual socket is used on the TRAMs themselves. Strips of 8 sockets are used on TRAM motherboards and as spacers (as in figure 48.8) between TRAMs and motherboards.

48.6.2 Through-board sockets

The component height given in figure 48.7 means that there is not enough height for conventional sockets for the components. A number of manufacturers make sockets which fit into a PCB in such a way that the thickness of the PCB is used for the socket, rather than extra height above the board.

INMOS has seen and used the following sockets. No particular recommendation for any of these is given or implied. Other manufacturers have shown data sheets for similar sockets with a height of approximately 0.8mm. The Augat 'Holite' sockets, which sit below the PCB surface, have been seen but not used. The

1. These parts are available from Scott Electronics Ltd, Tonbridge, Kent, England (Tel: 0372 359270), or Andon Electronics Corp, Albion, RI, USA (Tel: 401 333 0388).

Augat 'Solderite' sockets have similar dimensions to the Harwin 3153 and have been seen in prototype quantities. All of the sockets are available individually or assembled into strips; some are available in DIP and PGA format.

Manufacturer	type	height above PCB
Harwin (UK)	H 3153-01	0.38mm
Mark Eyelet (AMP) (US)	M8043PEC	0.2mm approx
PreciDIP (Switzerland)	014-92-001-41-012	0.4mm
Advanced Interconnections (US)	type -85	0.78mm
Harwin (UK)	H 3155-01	1.2mm
PreciDIP (Switzerland)	type 1407	0.8mm

48.6.3 Subsystem pins and sockets

The preferred socket to fit on the solder side of the TRAM is Harwin H 3153-01, and on the motherboard also. Samtec pin strip HLT-03-G-R is suitable for connecting between these sockets.

48.6.4 Motherboard sockets

The TRAM pins/stackable sockets will plug into any standard IC socket. To meet the component heights given in figure 48.7, the stackable socket (see section 48.6.1) must also be used on the motherboard.

Motherboard sockets for the Subsystem signals should be the 0.38mm or 0.4mm sockets referred to above.

48.7 Mechanical retention of TRAMs

Vibration tests have shown that in a normal office or laboratory environment, the TRAMs remain plugged into their sockets. In transit, however, or in an environment where there is vibration, some form of mechanical retention may be necessary.

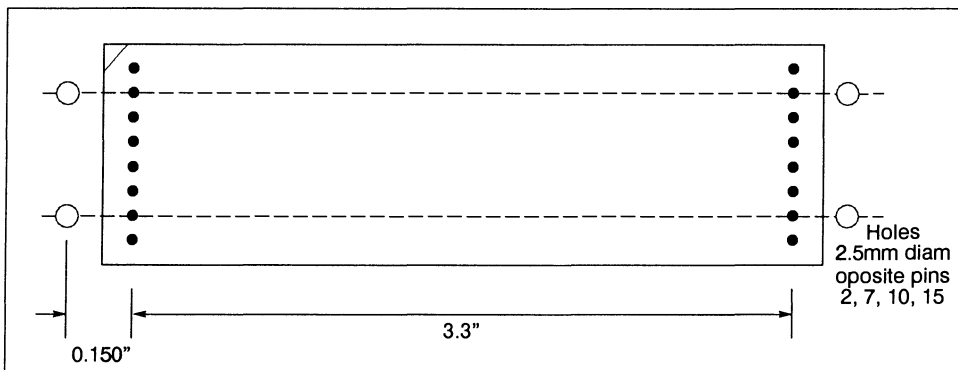


Figure 48.10 Fixing holes for mechanical retention

The detail drawings of the module sizes in section 48.8 show fixing holes in the modules. Similar fixing holes should be drilled in the motherboard as shown in figure 48.10. M2.5 nylon bolts may be used between these fixing holes to secure the modules.

48.8 Profile drawings

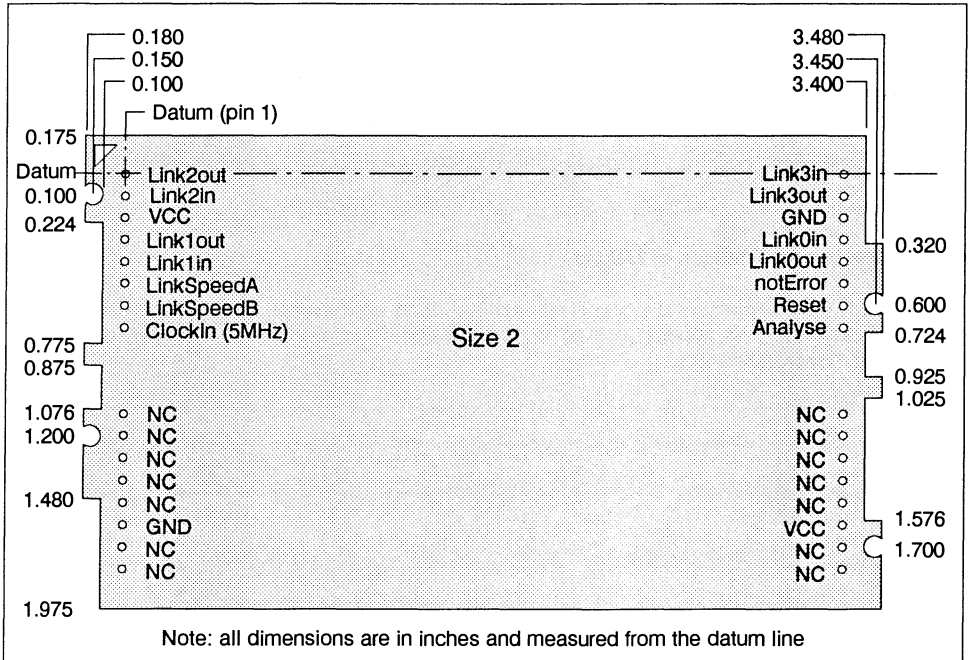
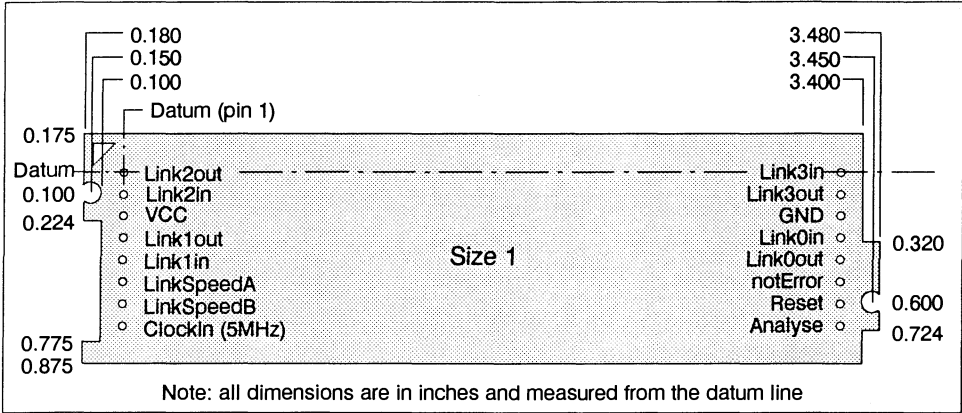


Figure 48.11 PCB profile drawings and pinout, TRAMs Sizes 1 and 2

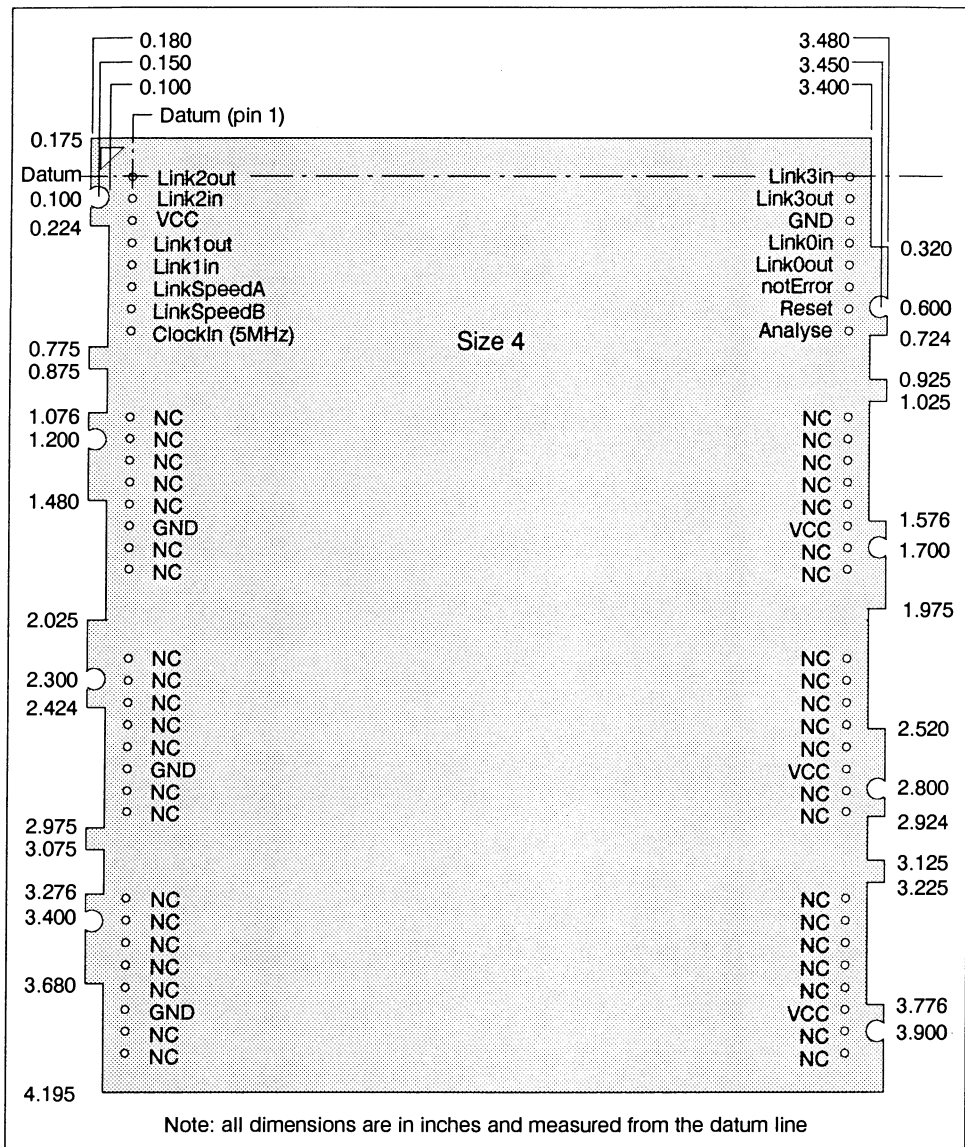


Figure 48.12 PCB profile drawings and pinout, TRAMs Size 4

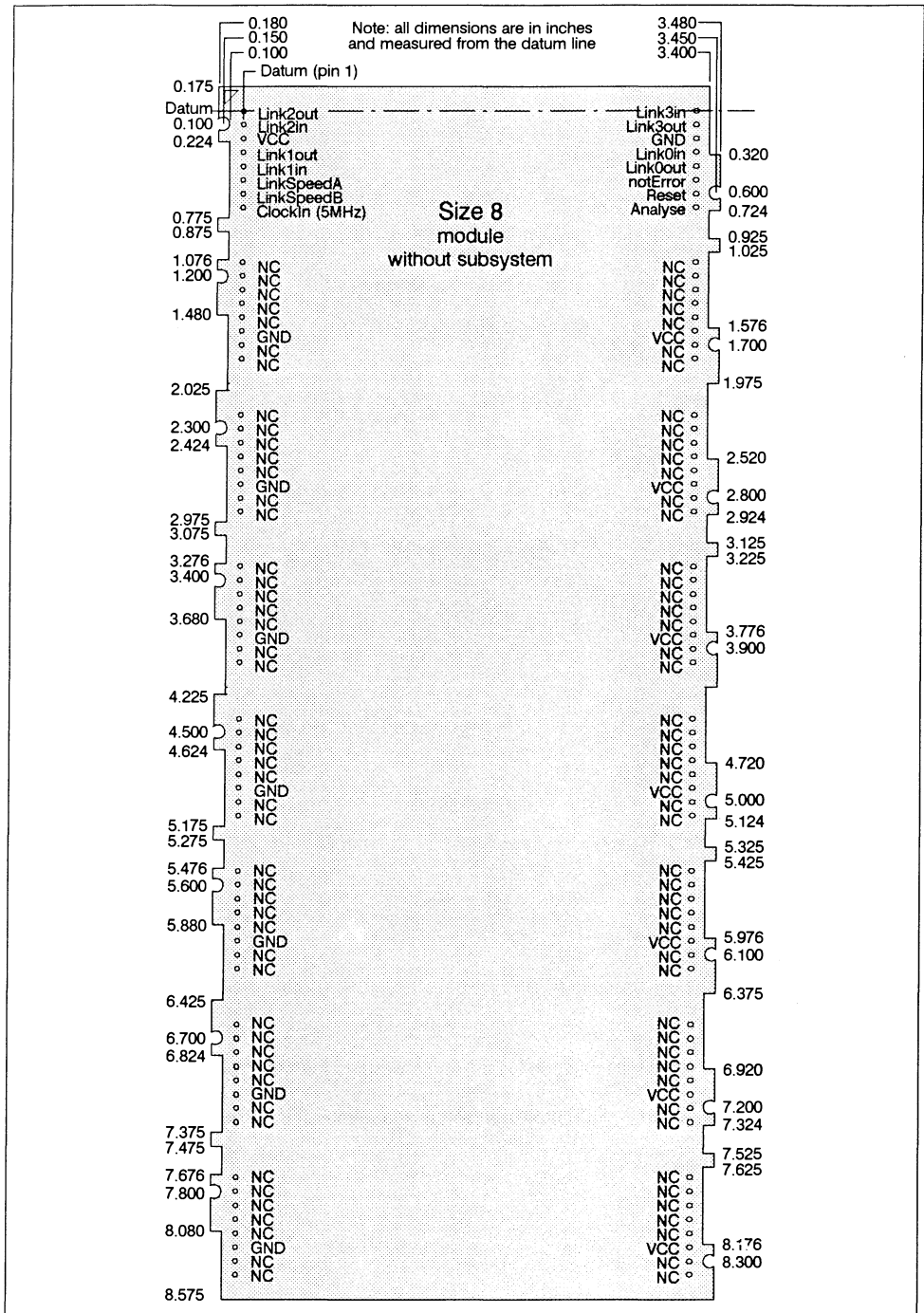


Figure 48.13 PCB profile drawing and pinout, TRAMs Size8 without subsystem

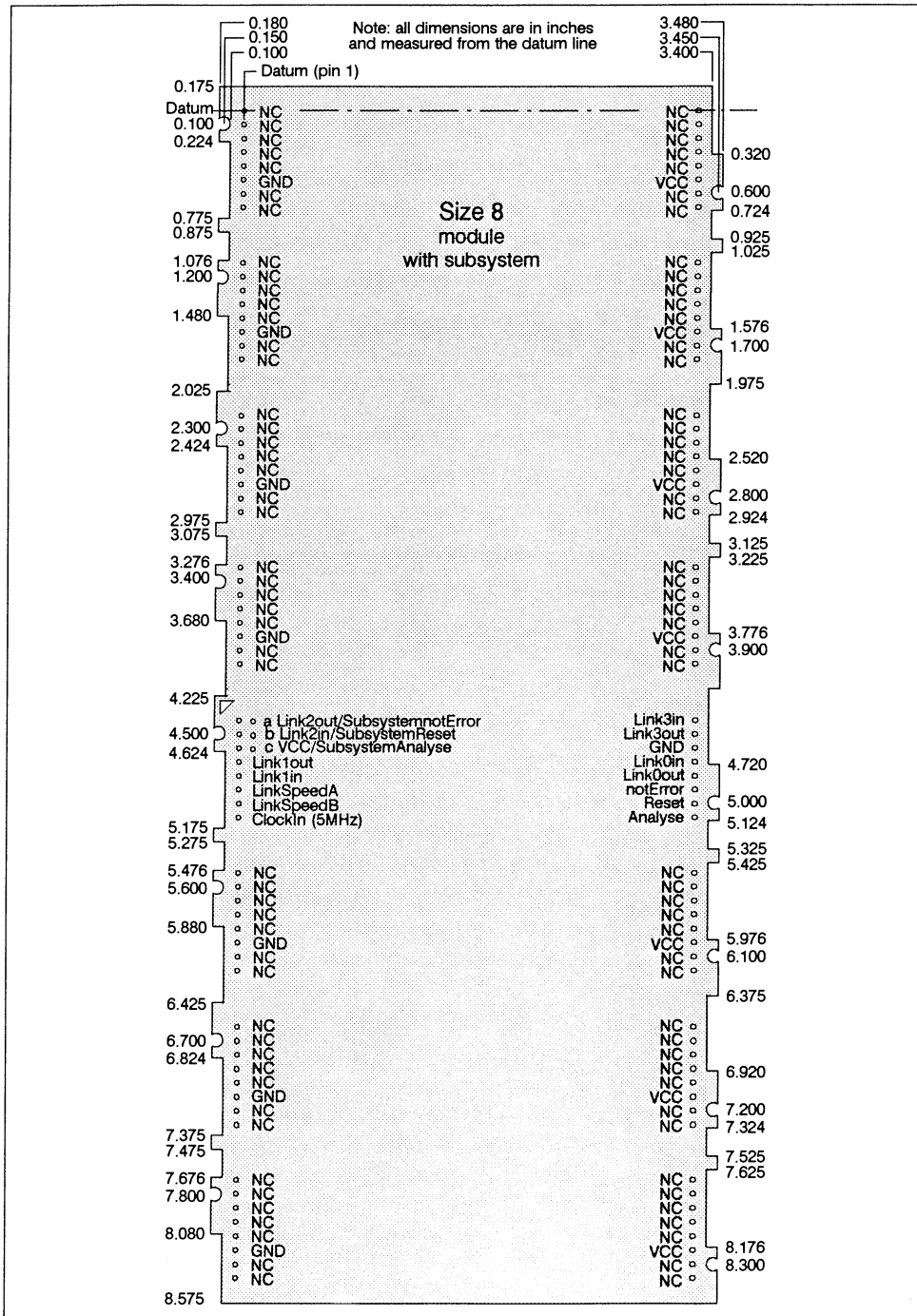
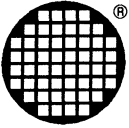


Figure 48.14 PCB profile drawing and pinout, TRAMs Size8 withsubsystem



Module Motherboard Architecture

(INMOS Technical Note 49)

49.1 Introduction

INMOS transputer modules are designed to form the building blocks of parallel processing systems. They consist of printed circuit boards in a range of sizes which typically hold a member of the transputer family of processors, some memory and perhaps some application specific circuitry. A module needs only a 5 volt power supply and a 5MHz clock to operate. These are supplied to the module through pins on the periphery of the board. Other pins bring out the transputer's serial links and reset, analyse and error signals. Some modules can control a subsystem of other modules through another set of pins. The *Dual-In-Line Transputer Modules (TRAMs)* document provides a complete specification of INMOS transputer modules.

In order to use modules as parallel processing building blocks INMOS has developed a range of motherboards. While these boards provide access to transputers from a number of different host machines, they have a common architecture to allow control and interconnection of potentially large numbers of transputers. This document describes the generic architecture of module motherboards. It is recommended that this specification is followed when designing in order to preserve compatibility with INMOS module motherboards.

49.2 Module motherboard architecture

The INMOS range of module motherboards has a common architecture making it easy to build and configure systems consisting of large numbers of transputer modules. The goals aimed at in the design of the module motherboards, and the architecture developed to achieve them, are described below.

49.2.1 Design goals

- To be able to build systems with any number of transputer modules in any combination of type or size
- To be able to build a variety of different kinds of network (e.g. arrays, trees, cubes, etc.)
- Enable any number of motherboards to be chained together
- Make transputer link connections easily configurable by software
- To be able to run test and applications programs on transputers without first configuring links
- Provide a standard hardware interface to configuration and applications software
- Allow hierarchical control of systems of transputers
- Make the transputer hardware and software independent of the host system

49.2.2 Architecture

In order to achieve the design goals outlined above, a standard architecture is adopted for all module motherboards. The rest of this document describes the motherboard architecture in detail, but the salient features are given below.

- The modules in a network are connected in a pipeline using two links from each module
- The remaining links from each module are taken to IMS C004 programmable link switches
- A number of links are taken from IMS C004s to edge connectors for wiring to other boards
- Each IMS C004 is controlled by an IMS T222 transputer
- The IMS T222s are connected in a separate pipeline

- The first module in the pipeline on a particular motherboard can control a subsystem of other transputers that may reside on the same motherboard, another motherboard or may be distributed across a number of boards
- An interface may be provided to enable a non-transputer based host system to control and communicate with a motherboard

49.3 Link configuration

Transputers communicate with each other via serial links operating at 10 or 20Mbits/s. The module motherboard architecture facilitates the interconnection of links between transputer modules by providing a standard hardware link configuration and allowing software configuration using IMS C004 programmable link switches. Links should be interconnected by properly terminated transmission lines (PCB trace or cable) having a characteristic impedance of 100Ω. INMOS Technical note 18, *Connecting INMOS links*, gives full details on all aspects of connecting links.

49.3.1 Pipeline

Each module resides in a *module slot* which provides two sockets that take the 16 pins of a size 1 module. A motherboard may have any number of module slots, determined only by the physical size of the board. The slots are numbered starting at slot 0.

All the modules on a motherboard are connected in a pipeline as shown in figure 49.1.

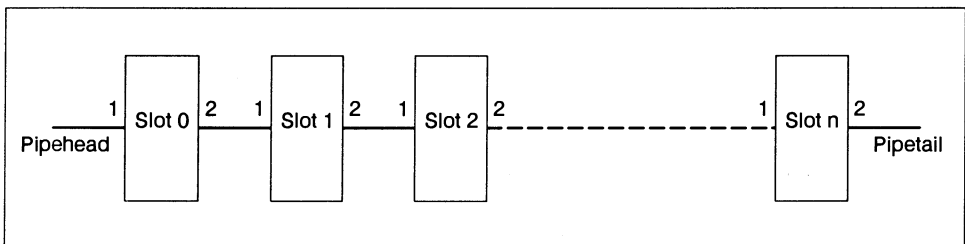


Figure 49.1 Module pipeline

Link 2 of the module in slot 0 is connected to link 1 of slot 1 and so on for the rest of the pipeline. Link 1 of module slot 0 (*Pipehead*) and link 2 of the last module slot (*Pipetail*) are brought out to an edge connector thus enabling the pipelines of any number of boards to be chained together by connecting *Pipehead* of one board to *Pipetail* of the next. See figure 49.2.

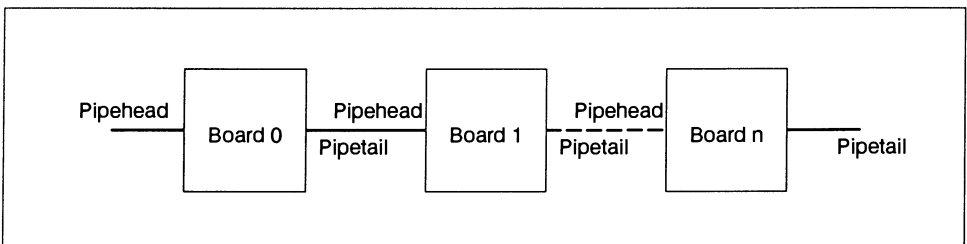


Figure 49.2 Module pipeline on several boards

Some applications may not require a full complement of modules or may use size 2 or larger modules which take up more than one slot, but use only one slot for electrical connection. In either case the pipeline will be broken unless steps are taken to keep it intact. A *pipe jumper* is a small connector used for this purpose. See figure 49.3. It plugs into an unused module slot and connects link 1 of that slot to link 2 of the same slot, thus preserving the pipeline.

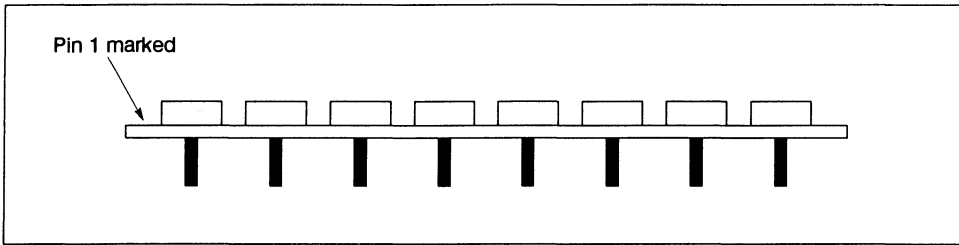


Figure 49.3 Pipe jumper

49.3.2 IMS C004 link configuration

An IMS C004 programmable link switch is used for software configuration of links. This device is a crossbar switch which can handle up to 32 links. It can connect any of the 32 link inputs to any of the 32 link outputs under software control from a separate configuration link.

Links 0 and 3 of each module are taken to an IMS C004 or a number of IMS C004s, depending on the number of links. Links may be taken from an IMS C004 to an edge connector to allow links from one motherboard to be connected to those of another.

The number of IMS C004s required on a particular motherboard depends on the number of modules the board can hold. The exact arrangement of IMS C004 links is not specified here in order to give the designer maximum flexibility for his particular application. **The only restriction is that links 0 and 3 of each module are taken to a C004.** This may be done in a number of ways. For example:

- Link 0s may be taken to one IMS C004 or a set of IMS C004s; link 3s may be taken to another IMS C004 or a set of them
- Both Link 0s and link 3s may be taken to the same IMS C004(s)
- LinkOut0s and LinkOut3s may be connected to an IMS C004 or a set of the same, while LinkIn0s and LinkIn3s are taken to another IMS C004 or a set of them

49.3.3 T222 pipeline and C004 control

Each IMS C004 on a motherboard is controlled from an IMS T222 16-bit transputer as shown in figure 49.4. An IMS T222 can control up to two IMS C004s via its links 0 and 3. Links 1 and 2 of each IMS T222 are used to connect the transputers in a *configuration pipeline*. Link 1 of the first IMS T222 on the board is taken to an edge connector designated *ConfigUp*; link 2 of the last IMS T222 in the board's configuration pipeline is also taken to an edge connector designated *ConfigDown*. In this way the configuration pipelines of any number of motherboards may be chained together by connecting *ConfigDown* of one board to *ConfigUp* of the next, enabling a network of transputer modules spread over several boards to be configured from software.

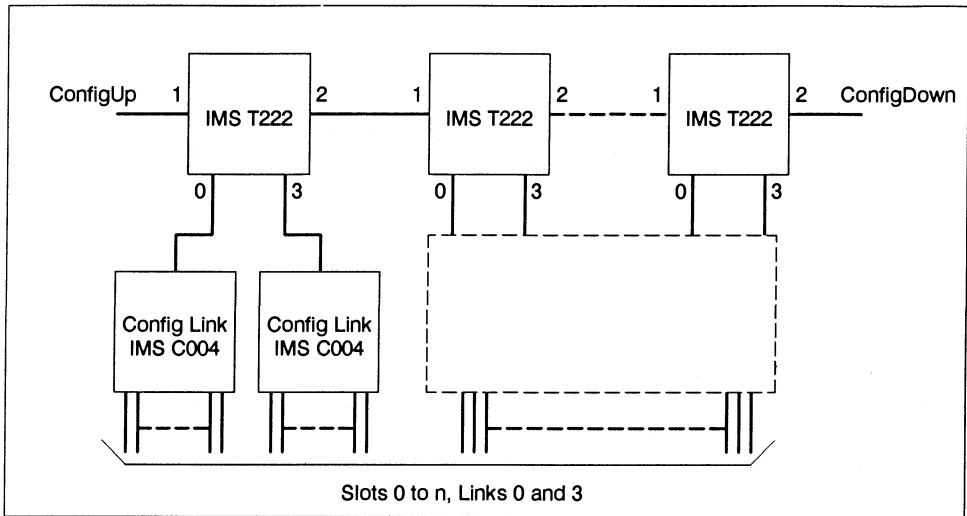


Figure 49.4 IMS C004 control by a pipeline of IMS T222s

The IMS C004 configuration data may come from software running on a module residing on the first motherboard in the system. It is therefore necessary to be able to connect a link of that module to the board's configuration pipeline. A jumper provides the option of connecting link 1 of the first IMS T222 in the configuration pipeline *either* to ConfigUp or to link 1 of module slot 0. In the latter, the jumper also disconnects PipeHead on the edge connector from slot 0 link 1. This is shown diagrammatically in figure 49.5.

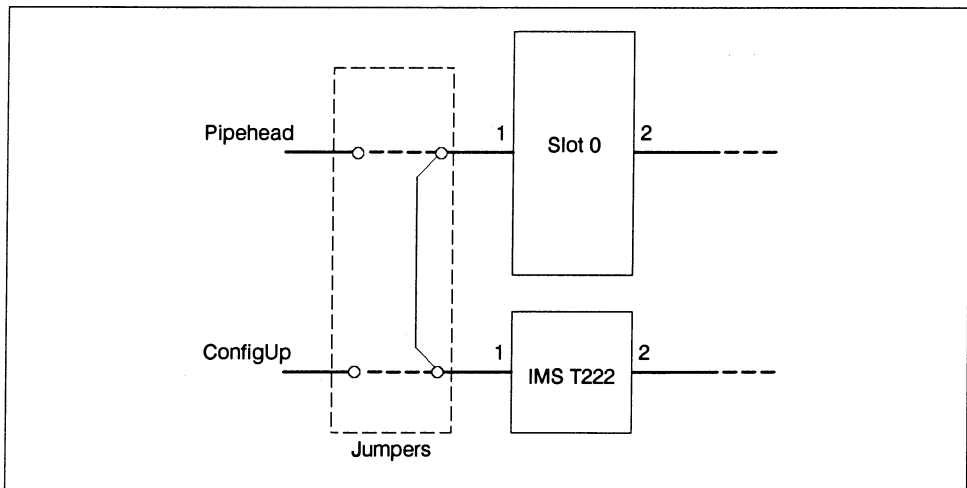


Figure 49.5 ConfigUp/Pipehead jumper

49.3.4 Software link configuration

The hardware configuration described in Sections 49.3.2 and 49.3.3 provides the standard architecture recognised by the *Module Motherboard Software (MMS)*, a software package available from INMOS which allows easy configuration of the IMS C004 link connections.

The MMS takes a list of link connections that are hardwired on the board together with a list of the required 'softwired' connections and generates the configuration details for each IMS C004.

For each board in the system, the user can:

- Connect link 0 of any module to link 3 of any module
- Connect link 0 or link 3 of any module to an edge connector link
- Connect an edge connector link to another edge connector link

The MMS is described in detail in the *MMS2 User Guide*.

49.4 System control

The subsystem control function of the module motherboard architecture allows hierarchical control of networks of transputers. It enables a module capable of driving a subsystem to reset or analyse a network of modules and to handle errors in the network. The driving module can itself form part of a network which is controlled by another module. In this way a hierarchy of control is made possible.

Each module on a motherboard requires a 5MHz clock. The module motherboard specification provides a scheme for distributing the clock signal from a single crystal oscillator to all the modules on a motherboard.

49.4.1 Reset, analyse and error

Three signals are provided by transputers for the purpose of allowing system control: *Reset*, *Analyse* and *Error*. The **Reset** and **Analyse** inputs enable the transputer to be initialised or halted in a way which preserves its state for subsequent analysis. The transputer **Error** signal is connected directly to the processor's Error flag. See the *Transputer Databook* for a detailed description of these signals.

A transputer module has a similar set of signals: module **Reset** and **Analyse** are connected directly to the respective pins on the transputer; the transputer **Error** pin is taken to a transistor on the module to produce an open collector **notError** signal that can be wire-ORed with the **notError** signals of other modules.

Some modules are capable of controlling a subsystem of other modules. They have three extra pins: **SubSystemReset**, **SubSystemAnalyse** and **notSubSystemError**, which are controlled by the on-module transputer through latches in memory. These pins are connected to the **Reset**, **Analyse** and **notError** pins of the modules in the subsystem being controlled. The subsystem can then be reset or analysed by asserting the relevant signal of the subsystem controller module. The subsystem's ORed **notError** signal can also be monitored by the controlling module.

49.4.2 Up, down and subsystem

A module motherboard has three ports that provide hierarchical control: Up, Down and subsystem (see figure 49.6). Each port appears at an edge connector and has three active-low signals: **notReset**, **notAnalyse** and **notError**. A board is able to control a subsystem of other boards by connecting its subsystem port to the Up port of the next board. Boards in a subsystem are chained together by connecting the Down port of one board to the Up port of the next board. A board within a subsystem is in turn able to control another network through its subsystem port.

Figure 49.7 shows how a board can be connected to a subsystem of boards.

The **notReset** and **notAnalyse** signals flow from subsystem of one board to Up of the next board. From there, they go directly to Down. They are also logical ORed with that board's subsystem reset and analyse latches and then pass to the subsystem port. The **notError** signal passes from a board through its Up port. If it is connected to the Down port of the board above, it is logical ORed with that board's Error signal and passed to the Up port. If it goes to the subsystem port of the board above, the Error signal is not passed

on, but is handled by that board. (Figures 49.10, 49.11 and 49.12 show the module motherboard system control logic.)

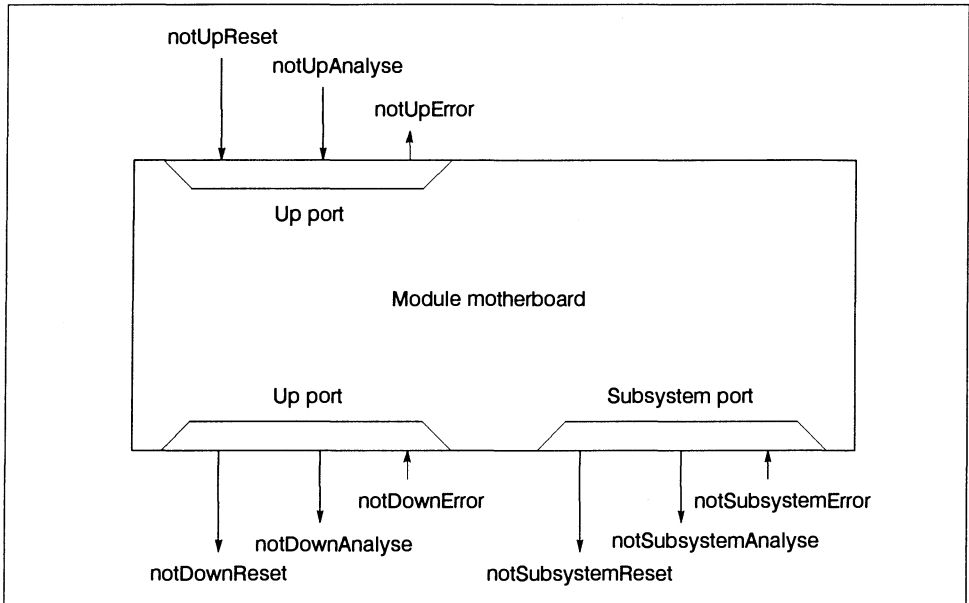


Figure 49.6 Up, down and subsystem

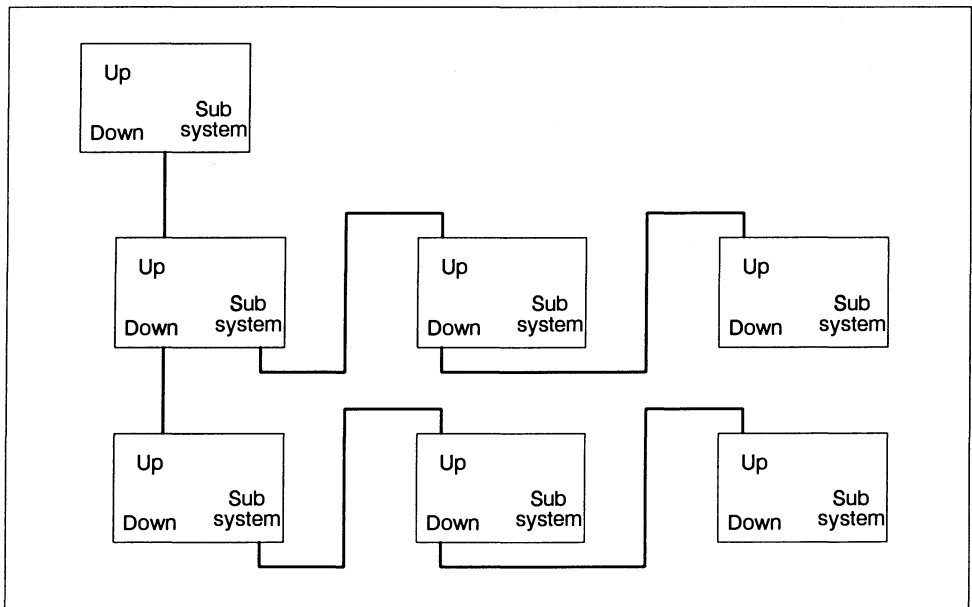


Figure 49.7 Controlling a subsystem of boards

49.4.3 Source of control

If there are n slots on a motherboard, modules in slots 1 to n may be controlled from either the Up port (or a host machine if the motherboard has an interface to one, see Section 49.5) or may be part of a subsystem controlled by a suitable module in slot 0. The source of control is determined by a jumper or switch, as shown in figure 49.8.

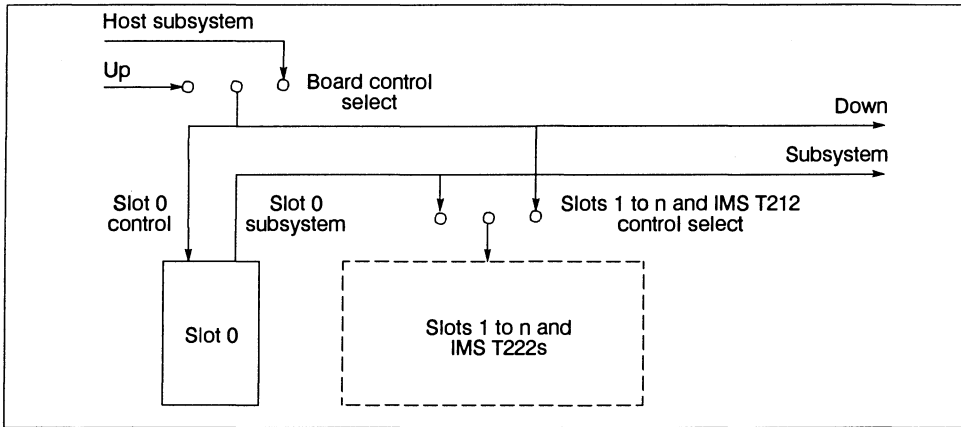


Figure 49.8 Source of control

The on-board IMS T222(s) may be reset and analysed from the same source that controls slots 1 to n . The **Error** pin of the IMS T222(s) is not connected.

A power-on reset circuit is required for the IMS C004(s) on board. An IMS C004 may then be reset at power-on or by the IMS T222 controlling it. Each IMS T222 has a latch mapped into its memory space. See figure 49.9. This enables software running on the IMS T222 to reset the IMS C004 either by setting the latch or by sending a reset message to the IMS C004 Configuration link.

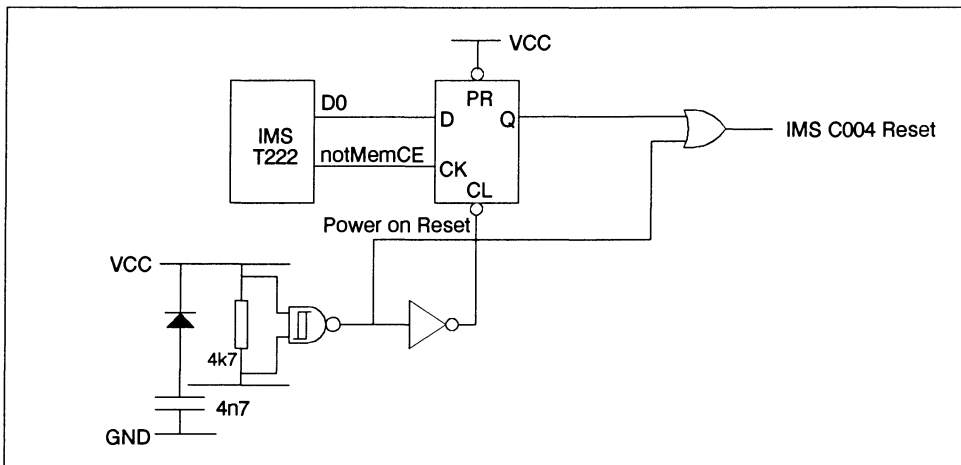


Figure 49.9 IMS C004 reset circuit

Figures 49.10, 49.11 and 49.12 show the logic required for Reset, IMS C004 Reset, Analyse and Error, respectively. These diagrams provide a logical description only: the actual implementation is left to the individual designer. It is important, however, to include the passive components indicated in the diagrams. The

1K pull-up resistors on the **notUpReset**, **notUpAnalyse**, **notDownError** and **notSubSystemError** signals are necessary to ensure that if these signals are unconnected they are not left floating, but are deasserted. The 4K7 pull-up resistors are required to wire-OR the open collector **notError** signals from the module slots. Note that the *Dual-In-Line Transputer Modules (TRAMs)* document specifies a maximum of ten **notError** signals should be wire-ORed together. The combination of each 100 Ω resistor and 100nF capacitor filters out noise on the **notUpReset**, **notUpAnalyse**, **notDownError** and **notSubSystemError** signals coming from off the board.

To improve noise rejection, it is recommended that Schmitt gates are used to receive signals from other boards. These gates should use bipolar technology (e.g low power Schottky 74LS series TTL). It is also recommended that gates driving signals off the board are capable of providing a full output voltage swing from 0V to 5V, e.g. HCT series gates.

The Reset logic (figure 49.10) uses the **Board Control Select** switch and multiplexer to select whether Slot 0 and the Down port are reset from the Up port or from the host. The **Slots 1 to n & IMS T222 Control Select** switch and multiplexer determine whether Slots 1 to n and the IMS T222s are reset from the Slot 0 subsystem port or from the Up port or the host. A similar arrangement is used for the Analyse logic (figure 49.11).

In the Error logic (figure 49.12), the **Slots 1 to n & IMS T222 Control Select** switches and multiplexers select whether **notError** from Slots 1 to n is passed either to the Slot 0 subsystem port or to the Up port or the host. The **Board Control Select** switch and decoder determine whether **Slots 1 to n notError**, **notDownError** or **notSlot0Error** are passed to the Up port or to the host.

Board Control Select and **Slots 1 to n & IMS T222 Control Select** correspond to the conceptual switches in figure 49.8.

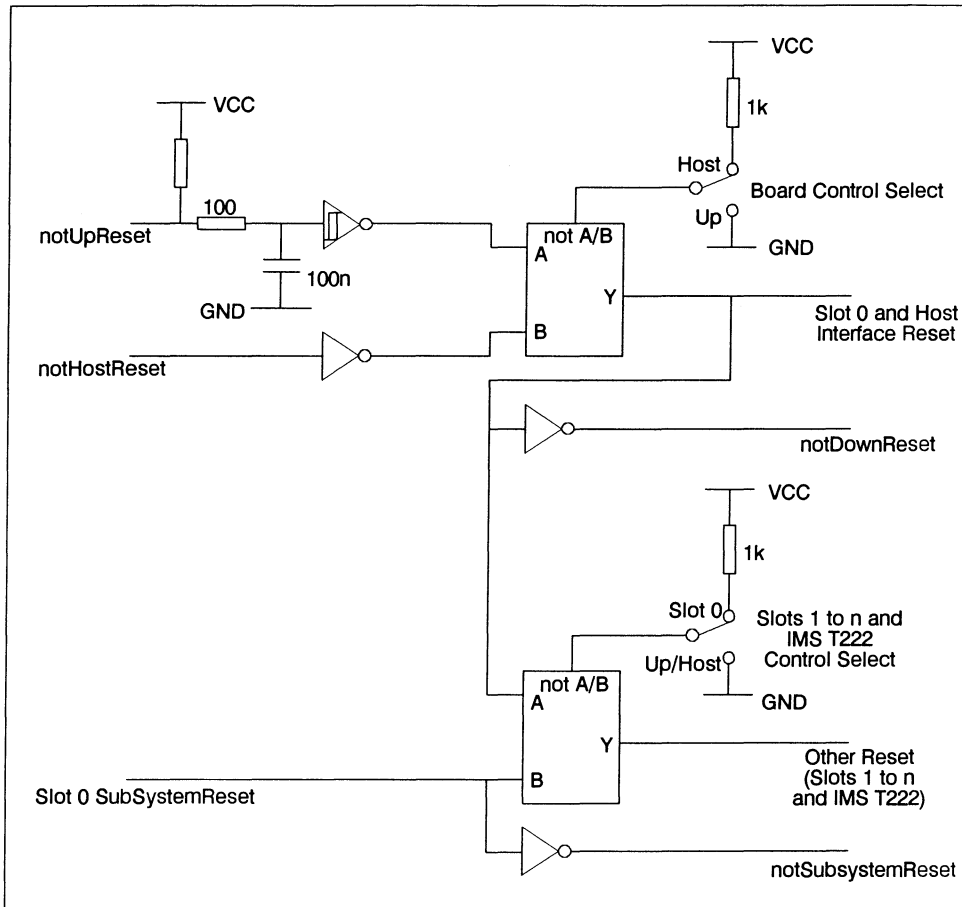


Figure 49.10 Reset logic

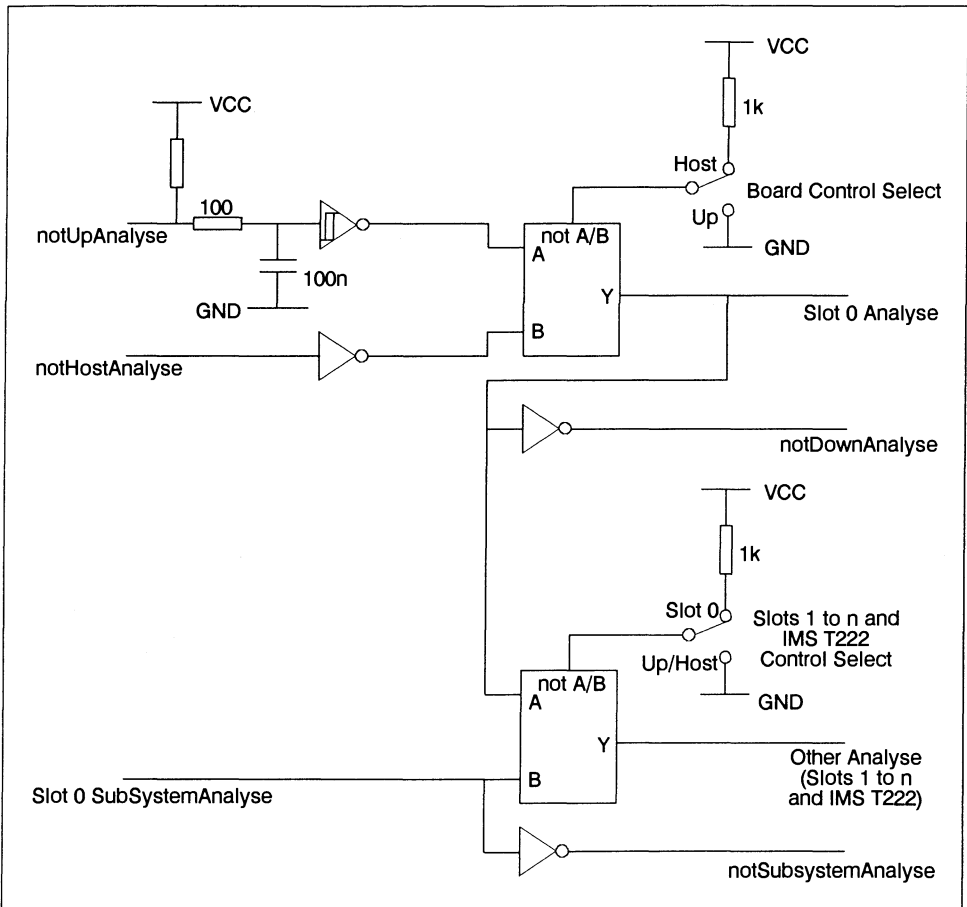


Figure 49.11 Analyse logic

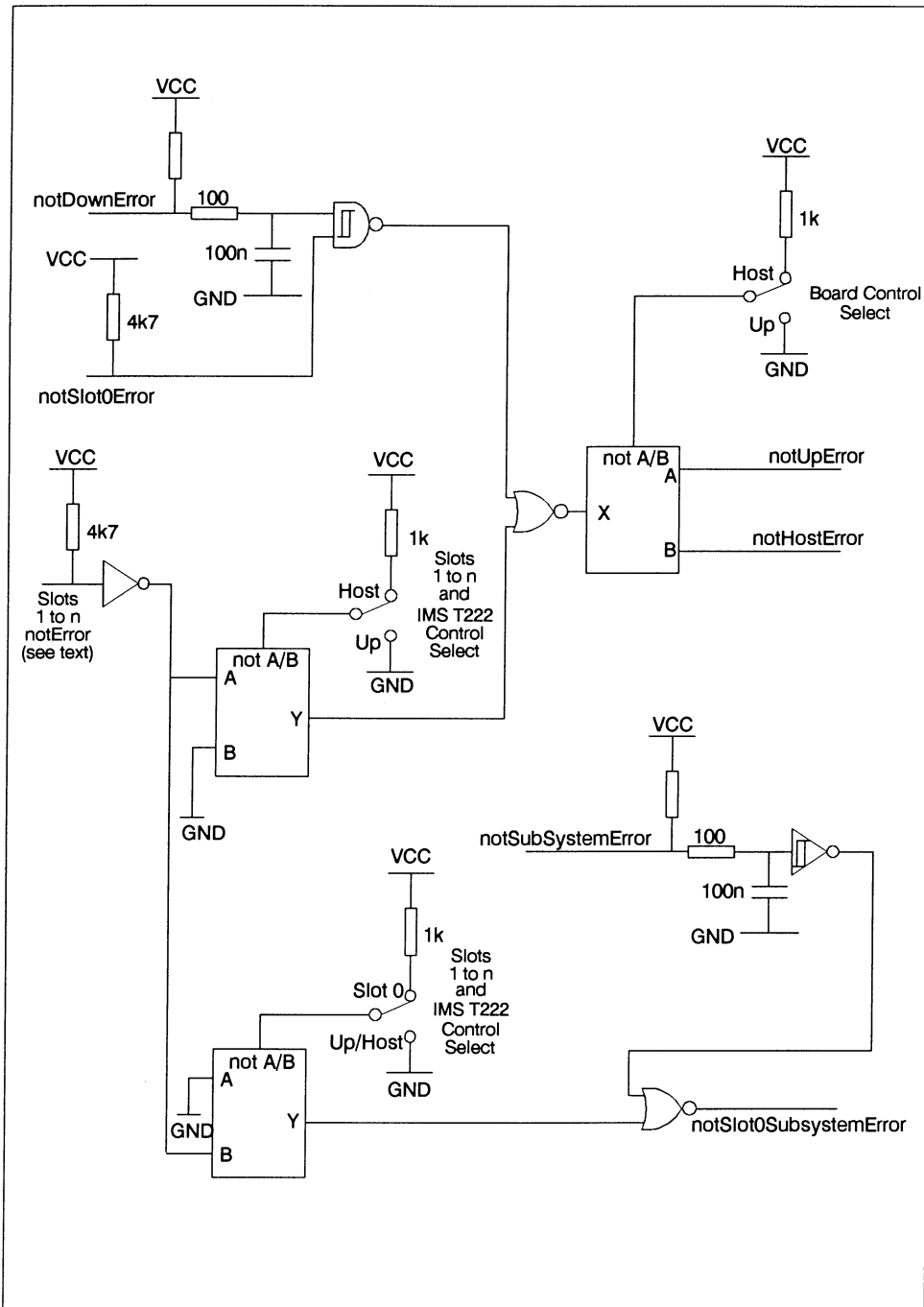


Figure 49.12 Error logic

49.4.4 Clock

A 5MHz, TTL compatible clock signal is required for each module slot, IMS T222 and IMS C004 on board. Since the clock must be distributed to a number of modules and devices the buffering scheme shown in figure 49.13 is used to minimise distortion of the clock waveform caused by excessive loading and transmission line effects. This is a *star* configuration and it may be extended indefinitely by adding more buffers at the star points which may drive further buffers, and so on until the required number of clock signals are derived. The length of any pcb trace carrying a clock signal should be limited to 30cm.

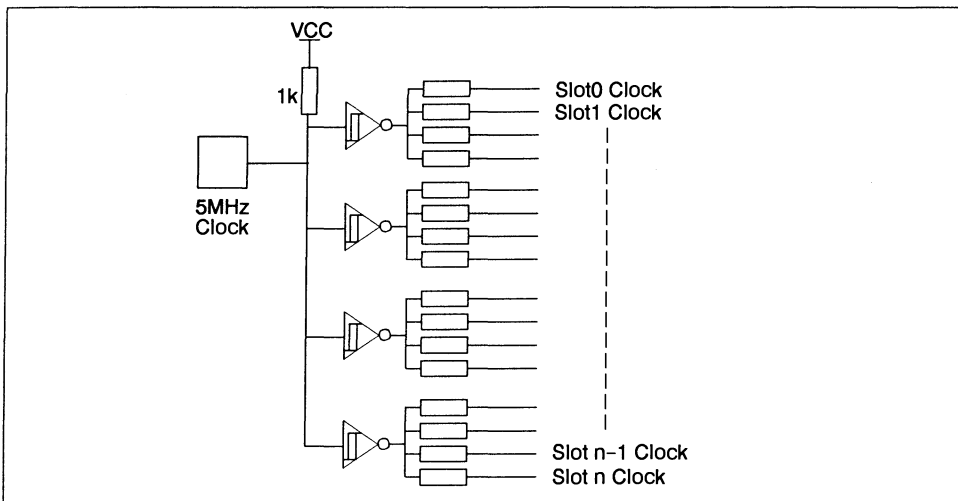


Figure 49.13 Clock distribution

49.5 Interface to a separate host

Some module motherboards may require an interface to a host machine or system that is not transputer based, e.g. the IBM PC, VMEbus or Futurebus. Because the implementation of the interface is specific to the host system, it is not defined here. However, it should allow the system to access the module pipeline and control a subsystem of modules.

49.5.1 Link Interface

The host system accesses the module pipeline via Slot 0 Link 0, as shown in figure 49.14. It is beyond the scope of this document to define the implementation of the host to link interface, but it might consist of an INMOS link adapter, the registers of which may be mapped into the host's address space, or it may involve the use of dual-ported RAM shared between the host and a transputer.

The interface must be capable of interrupting the host when a data transfer in either direction has been completed.

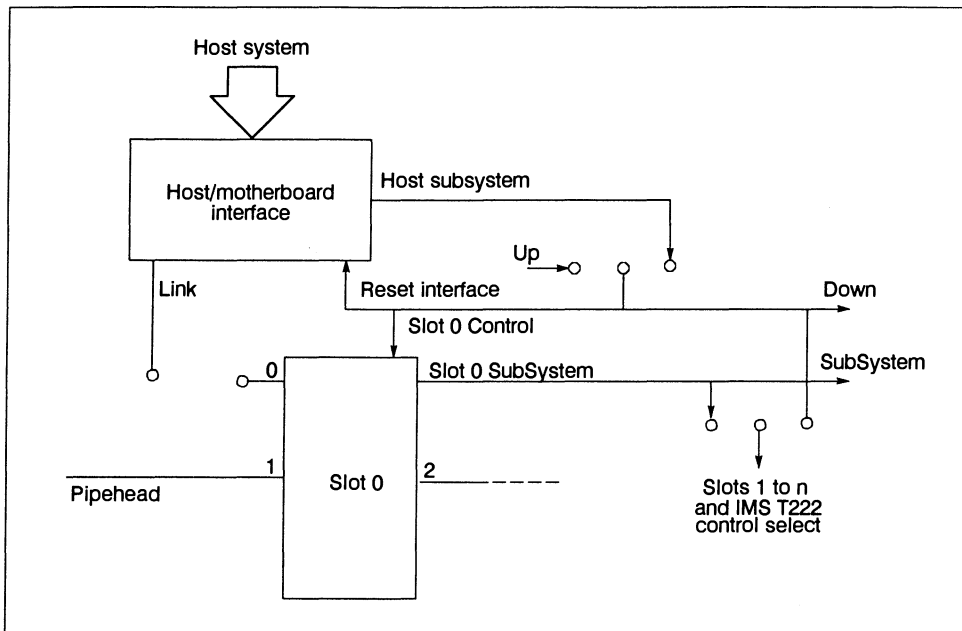


Figure 49.14 Host to motherboard interface

49.5.2 System control interface

The host system must be able to control a network of modules. This is made possible by the provision of latches mapped into the host's memory. There are three latches: Reset, Analyse and Error, which correspond to the **notHostReset**, **notHostAnalyse** and **notHostError** signals of the *HostSubSystem* port shown in figure 49.14. The Reset and Analyse latches are mapped into successive locations of host memory. Reset and Analyse are write only by the host; the Error latch is read only and shares the same address as the Reset latch.

Writing a '1' into bit 0 of the Reset latch asserts **notHostReset**;
Writing a '0' into bit 0 of the Reset latch deasserts **notHostReset**.

Writing a '1' into bit 0 of the Analyse latch asserts **notHostAnalyse**;
Writing a '0' into bit 0 of the Analyse latch deasserts **notHostAnalyse**.

A '1' read in bit 0 of the Error latch indicates that **notHostError** is asserted;
A '0' read in bit 0 of the Error latch indicates that **notHostError** is deasserted.

The host to motherboard link interface is reset by the same source as Slot 0, i.e. the Up port or the HostSubSystem port.

49.5.3 Interrupts

The host to subsystem interface must be capable of generating an interrupt to the host when certain events occur on the motherboard. These include:

- Completion of transfer of data from the host to the motherboard
- Completion of transfer of data from the motherboard to the host
- Error in subsystem indicated by **notHostError** being set

Other system specific conditions may also generate an interrupt, e.g. if DMA is used to transfer data between the host and motherboard, the end of a DMA cycle may trigger an interrupt.

The host may select which conditions cause an interrupt by setting bits in a register or registers on the motherboard, mapped into the address space of the host. Other registers hold status information that can be read by the host to determine the source of an interrupt.

49.6 Mechanical considerations

The size and shape of a module motherboard is determined by its application. However, there are a number of mechanical constraints which must be adhered to in order to maintain compatibility between different modules and motherboards.

The size and spacing of module slots must conform to the mechanical specification in the *Dual-In-Line Transputer Modules (TRAMs)* document, the main points of which are reiterated here.

49.6.1 Dimensions

In the following, dimensions are quoted in inches for PCB length, width and related dimensions; all other dimensions are quoted in millimetres.

Width and length

The basic size of a TRAM is a very wide 16 pin DIP, with 3.3" between the two rows of pins. These TRAMs fit on a 3.6" pitch on their length, and a 1.1" pitch on their width. Extra length is added beyond the pins to hold the pins, to provide for mechanical fixing, and to polarise the module shape. Modules can be made larger than the standard size by keeping the 3.3" between pins and using two or more sets of the 16 pins. They can be made smaller than the standard size, down to a 16 pin DIP with 0.6" between the two rows of pins, or 1.5" between the pins. These sizes will normally be used for single chip modules or hybrids.

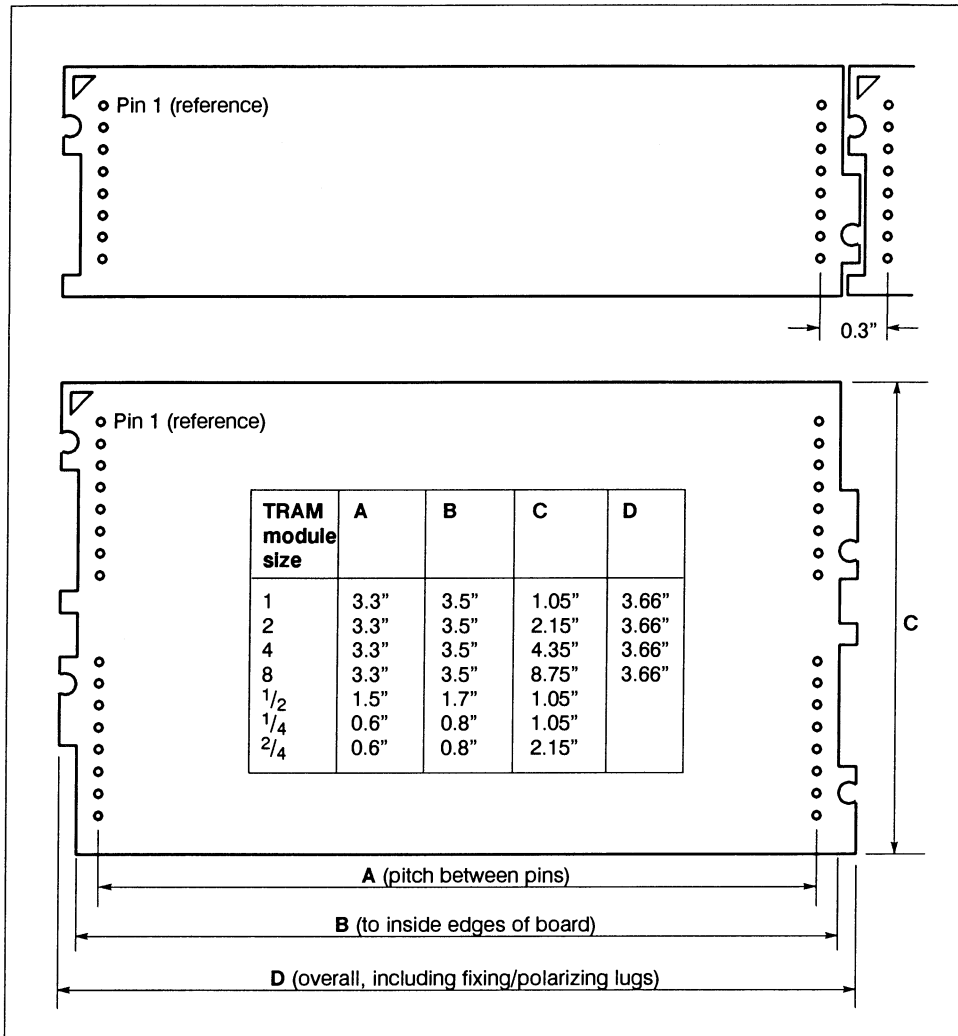


Figure 49.15 Transputer module sizes

The top drawing in figure 49.15 shows a Size1 module and how the jigsaw pattern fits together between adjacent modules. The lower drawing in figure 49.15 shows the various sizes of TRAM. Detailed dimensions of the different sizes are given in the *Dual-In-Line Transputer Modules (TRAMs)* document.

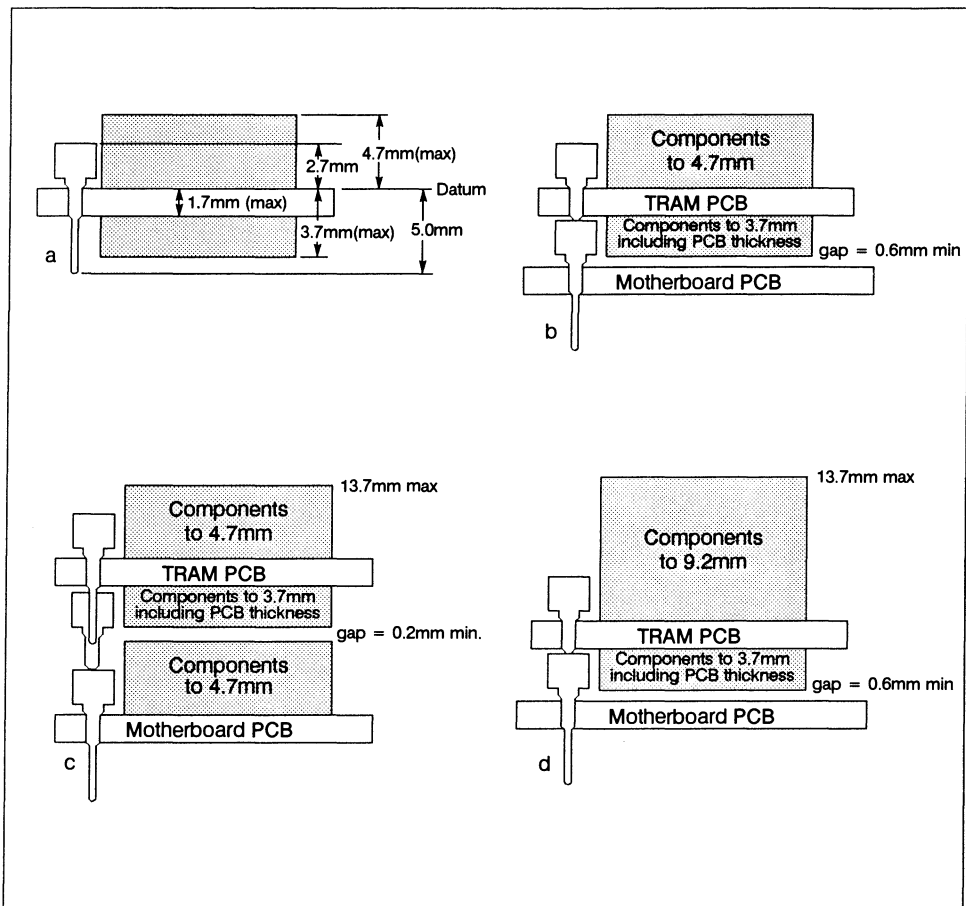


Figure 49.16 Component heights

Vertical dimensions

The height specifications, both above and below the TRAM PCB, are shown in figure 49.16a. Figure 49.16b shows a module with these dimensions plugged into a motherboard.

Figure 49.16c shows a TRAM above components on a motherboard and the overall component height is 13.7mm, which is within normal specifications for motherboards on 0.8" centres.

It is recommended that any component reaching a maximum specified height has an insulating surface.

To provide the spacing shown in figure 49.16c, the TRAM pins are implemented as a stackable socket, and an extra stackable socket is used between the motherboard socket and module pin.

Figure 49.16d shows an alternative component height which meets the 13.7mm overall height if the module is not above components on a motherboard.

Note that the datum for component heights on both sides of the TRAM is the component side surface. This datum is also used for the stackable socket to minimize tolerance buildup.

49.6.2 Motherboard sockets

The TRAM pins/stackable sockets defined in the *Dual-In-Line Transputer Modules (TRAMs)* document will plug into any standard IC socket. To meet the component heights given in figure 49.16, the stackable socket must also be used on the motherboard.

Motherboard sockets for the Slot 0 subsystem signals should be the 0.38mm or 0.4mm sockets referred to in the *Dual-In-Line Transputer Modules (TRAMs)* document.

49.6.3 Mechanical retention of TRAMs

Vibration tests have shown that in a normal office or laboratory environment, the TRAMs remain plugged into their sockets. In transit, however, or in an environment where there is vibration, some form of mechanical retention may be necessary.

Modules have fixing holes to facilitate mechanical retention, see the *Dual-In-Line Transputer Modules (TRAMs)* document. Similar fixing holes should be drilled in the motherboard as shown in figure 49.17. M2.5 nylon bolts may be used between these fixing holes to secure the modules.

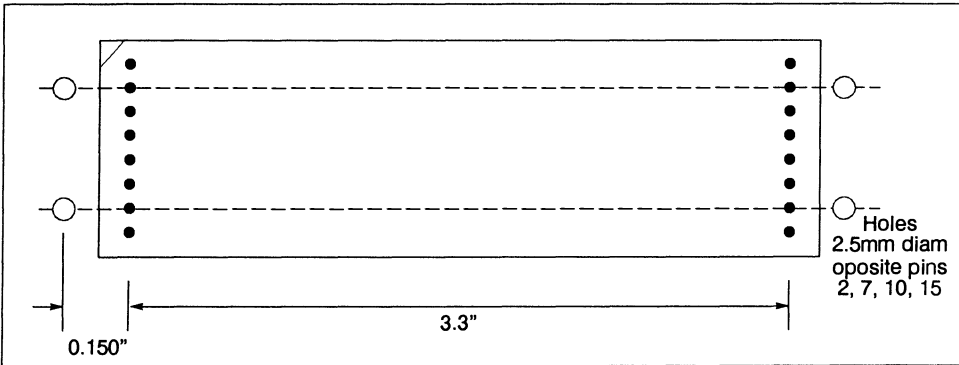


Figure 49.17 Fixing holes for mechanical retention

49.6.4 Module orientation

Figure 49.18 shows the orientation of transputer modules when mounted in slots on a motherboard. Notice how each module is rotated through 180° with respect to adjacent modules. This serves two purposes: cooling of Size 1 modules is improved; and it makes it possible to have Single-In-Line modules at some future date.

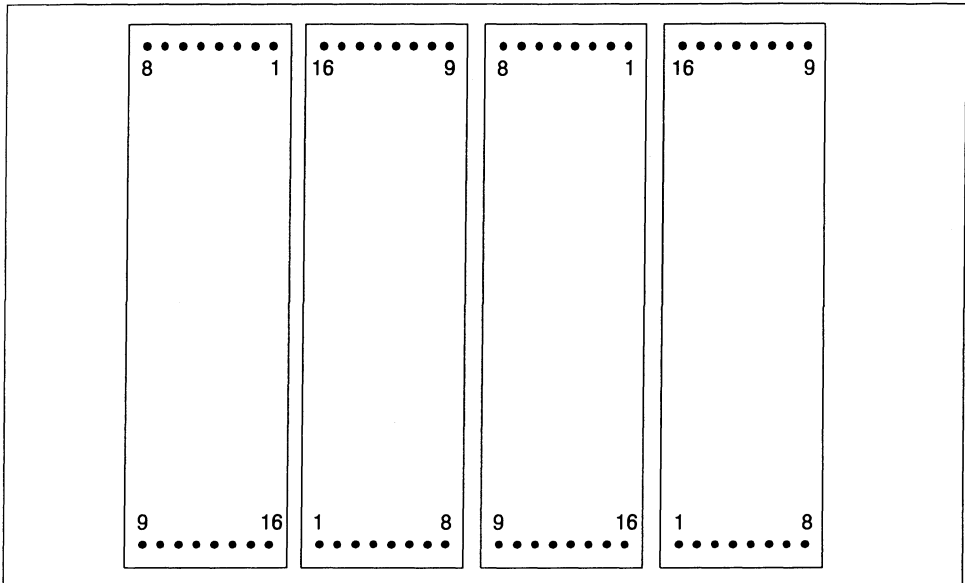


Figure 49.18 Orientation of module slots

49.7 Edge connectors

Connectors are necessary to enable links and system control signals to be taken from a motherboard to other boards. Several types of connector have been used on INMOS module motherboards.

The IMS B008 module motherboard for the IBM PC uses a 37-way D-type connector, the pin-out of which is shown in figure 49.19.

notUpReset	20	1	GND
notUpAnalyse	21	2	notUpAnalyse
notUpError	22	3	EdgeLinkOut0
EdgeLinkIn0	23	4	EdgeLinkIn1
EdgeLinkIn1	24	5	GND
EdgeLinkOut2	25	6	EdgeLinkIn2
EdgeLinkOut3	26	7	EdgeLinkIn3
EdgeLinkOut4	27	8	EdgeLinkIn4
GND	28	9	EdgeLinkOut5
EdgeLinkIn5	29	10	EdgeLinkOut6
EdgeLinkIn6	30	11	EdgeLinkOut7
EdgeLinkIn7	31	12	GND
ConfigUpLinkOut	32	13	ConfigUpLinkIn
PipeHeadLinkOut	33	14	PipeHeadLinkIn
notSubSystemReset	34	15	notSubSystemAnalyse
notSubsystemError	35	16	PipeTailLinkOut
PipeTailLinkIn	36	17	ConfigDownLinkOut
ConfigDownLinkIn	37	18	notDownReset
notDownAnalyse		19	notDownError

Figure 49.19 37-way D-type connector

This connector provides up to twelve links (including ConfigUp, ConfigDown, PipeHead and PipeTail), plus Up, Down and Subsystem ports. A cable suitable for connecting IMS B008s together is shown diagrammatically in figure 49.20.

The IMS B012 is a module motherboard in double extended Eurocard format. It has two 96-way DIN 41612 connectors. The bottom connector (P2) provides connections for eight links (including ConfigUp, ConfigDown, PipeHead and PipeTail) and Up, Down and SubSystem ports. Table 49.1 shows the general pinout adopted by INMOS for such a connector, making it suitable for use with module motherboards while preserving compatibility with the rest of the INMOS range of boards. The pins marked *Spare* and *Spare link* may be used for signals and links specific to a particular application. The *IMS B012 User Guide and Reference Manual* describes how these pins are used on the IMS B012.

The top connector (P1) of the IMS B012 is a DIN 41612 connector that takes a special mini-backplane to provide connections to 32 links. See figure 49.21 for the mechanical details and Table 49.2 for the pinout of this connector. On the IMS B012, the P1 connector is used to bring out links from the board's two IMS C004s. See the *IMS B012 User Guide and Reference Manual* for details. The mini-backplane is available from Varelco, part number 07-8258-0940-01-00. Both the P1 and P2 connectors are used with the INMOS Link and Reset cables provided with most INMOS board products.

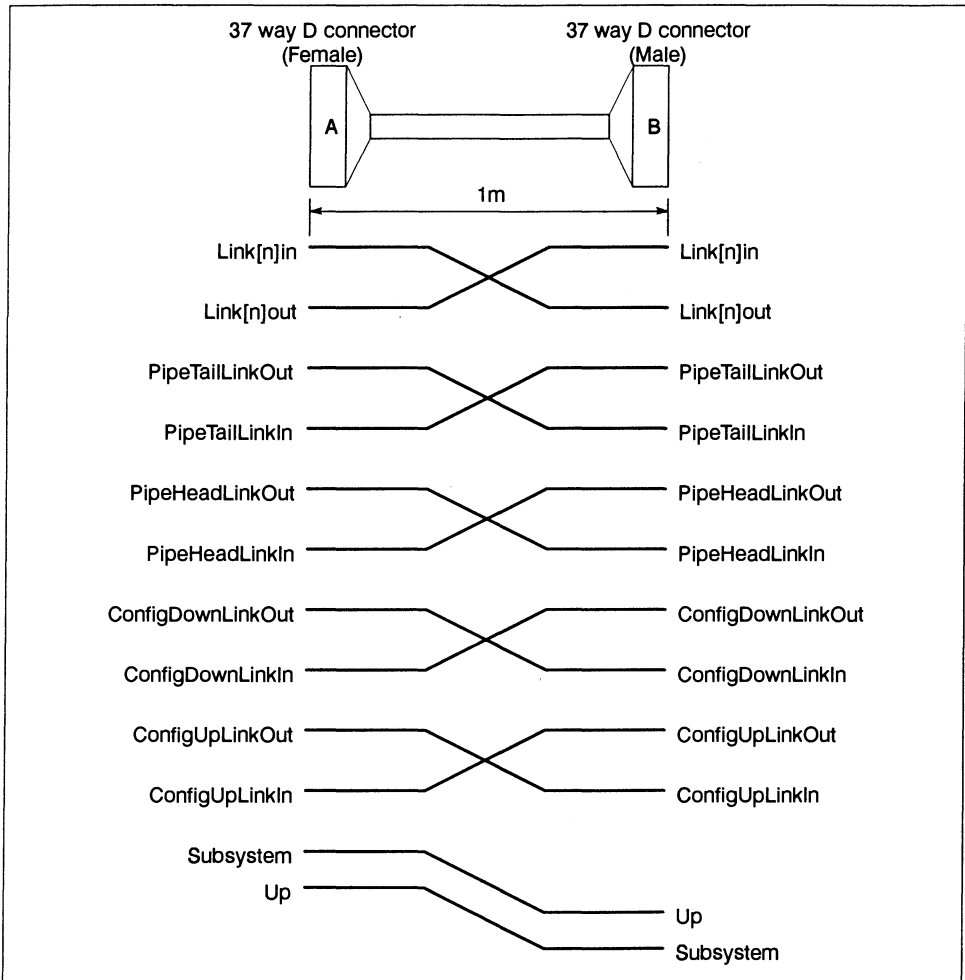


Figure 49.20 37-way cable

	c	b	a
1	GND	GND	GND
2	VCC	VCC	VCC
3	PAUX	nc	PAUX
4	VCC	VCC	VCC
5	GND	GND	GND
6	VCC	VCC	VCC
7	GND	GND	GND
8	nc	nc	nc
9	PipeHeadOut	Spare linkout	PipeTailOut
10	PipeHeadIn	Spare linkin	PipeTailIn
11	GND	GND	GND
12	nc	nc	nc
13	GND	GND	GND
14	nc	nc	nc
15	ConfigUpOut	Spare linkout	ConfigDownOut
16	ConfigUpIn	Spare linkin	ConfigDownIn
17	GND	GND	GND
18	nc	nc	nc
19	Spare	nc	Spare
20	Spare	nc	nc
21	Spare	GND	nc
22	Spare	nc	notSubReset
23	Spare	Spare linkout	notSubAnalyse
24	Spare	Spare linkin	notSubError
25	Spare	GND	GND
26	Spare	nc	nc
27	nc	GND	nc
28	notUpReset	nc	notDownReset
29	notUpAnalyse	Spare linkout	notDownAnalyse
30	notUpError	Spare linkin	notDownErro
31	GND	GND	GND
32	GND	GND	GND

Table 49.1 P2 DIN 41612 connector pin out

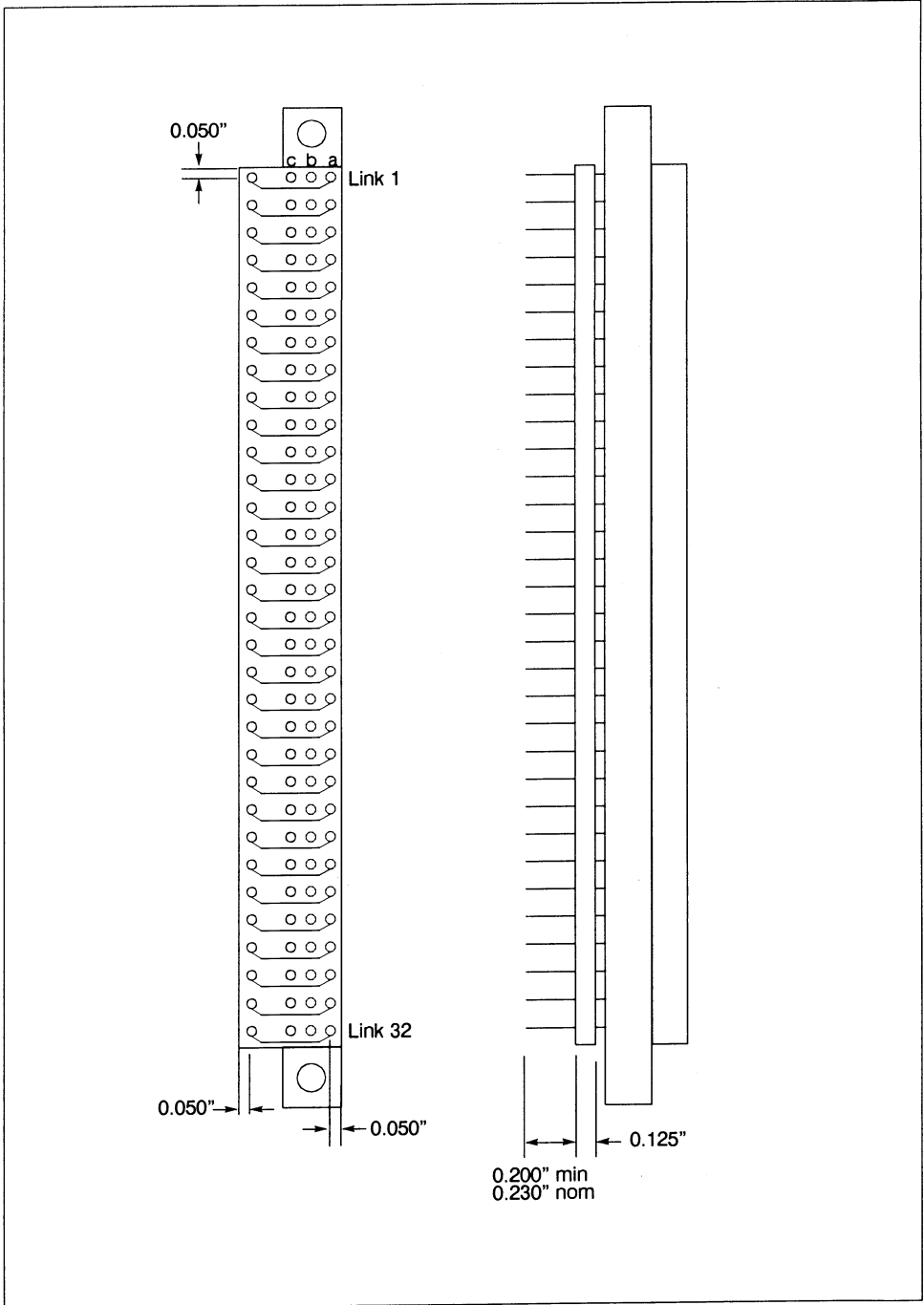


Figure 49.21 P1 32-link connector

	c	b	a
1	LinkOut0	LinkIn0	GND
2	LinkOut1	LinkIn1	GND
3	LinkOut2	LinkIn2	GND
4	LinkOut3	LinkIn3	GND
5	LinkOut4	LinkIn4	GND
6	LinkOut5	LinkIn5	GND
7	LinkOut6	LinkIn6	GND
8	LinkOut7	LinkIn7	GND
9	LinkOut8	LinkIn8	GND
10	LinkOut9	LinkIn9	GND
11	LinkOut10	LinkIn10	GND
12	LinkOut11	LinkIn11	GND
13	LinkOut12	LinkIn12	GND
14	LinkOut13	LinkIn13	GND
15	LinkOut14	LinkIn14	GND
16	LinkOut15	LinkIn15	GND
17	LinkOut16	LinkIn16	GND
18	LinkOut17	LinkIn17	GND
19	LinkOut18	LinkIn18	GND
20	LinkOut19	LinkIn19	GND
21	LinkOut20	LinkIn20	GND
22	LinkOut21	LinkIn21	GND
23	LinkOut22	LinkIn22	GND
24	LinkOut23	LinkIn23	GND
25	LinkOut24	LinkIn24	GND
26	LinkOut25	LinkIn25	GND
27	LinkOut26	LinkIn26	GND
28	LinkOut27	LinkIn27	GND
29	LinkOut28	LinkIn28	GND
30	LinkOut29	LinkIn29	GND
31	LinkOut30	LinkIn30	GND
32	LinkOut31	LinkIn31	GND

Table 49.2 P1 DIN 41612 connector pin out



Developing parallel C programs for transputers

(INMOS Technical Note 68)

50.1 Introduction

This document presents a cook book approach to writing parallel C programs for single and networks of transputers.

Using the **IMS Dx214** series C toolset the user can write programs which contain many parallel processes, these *processes* can then be mapped onto a number of transputers using a method called *configuration*.

Although this technical note is aimed primarily at new users to the INMOS development systems, advanced users may find this useful to come quickly up to speed with the new **IMS Dx214 C** toolset. Also, users of the **IMS D711 3L/INMOS** toolset may find this note of interest as there is a section on how to convert existing **IMS D711 C** code over to the **IMS Dx214** toolset.

All of the examples presented were developed using the following equipment:

- **IMS D7214 C** Toolset for IBM-PC.
- **IMS B008** Motherboard for IBM-PC.
- **IMS B404 TRAM** (Two were required for the network example).

50.2 What is the C toolset?

50.2.1 Introduction

The **IMS Dx214** is a software cross development system for transputers, hosted on a variety of platforms e.g. PC, SUN3, SUN4 or VAX. The development system consists of a set of tools to enable users to write programs for single or multiple transputer networks.

In this section we shall look at the *typical* development cycle for code running on single or multiple transputers. A brief outline of each tool used at each stage is given. Later on in the document we shall show, by means of worked examples, exactly how to use the tools at each step. For now this serves as a guide to where the tools fit into the grand scheme of things!

50.2.2 Software toolset summary

The following is a brief list of all the tools provided in the toolset.

Tool	Description
icc	The ANSI C compiler.
icconf	The configurer.
icollect	The code collector.
icvlink	The TCOFF file convertor.
idebug	The network debugger.
idump	The memory dumper. Used by idebug.
iemit	The transputer memory configuration tool.
ieprom	The EPROM program formatter tool.
ilibr	The toolset librarian.
ilink	The toolset linker.
ilist	The binary lister.
imakef	The Makefile generator.
iserver	The host file server.
isim	The IMS T425 simulator.
iskip	The skip loader tool.

Table 50.1 Summary of toolset components

50.2.3 Software design cycle – single transputer systems

This section will take a look at the major tools and steps involved in developing software for a *single transputer* system.

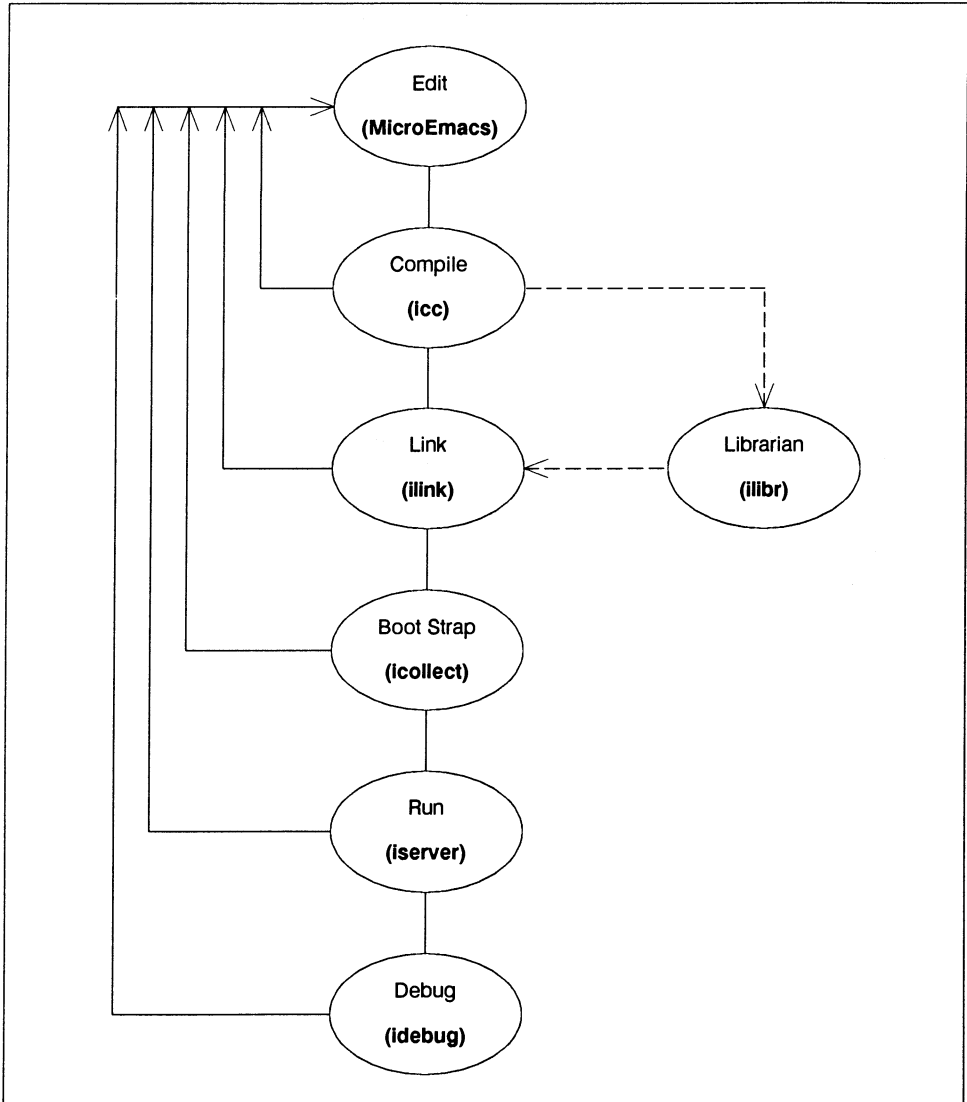


Figure 50.1 Typical software development route – single transputer systems

Lets take a look at each of the steps:

Edit

The `edit` phase consists of writing the source code for your program, this will include all of the source modules and header files. Any standard text editor can be used for the C Toolset, e.g. **MicroEmacs**, **Microsoft Word** and even `edlin`.

Compile

The compilation phase consists of submitting the source code to the compiler. The compiler is called `icc`, which stands for **Inmos C Compiler**. It requires the name of the source file and which processor type you wish to compile to.

Linking

Linking is achieved using the `ilink` tool. This pulls together all of the object files and any libraries that have been created. When compiling for a single transputer then the whole of the C Run Time library is used to give access to the host services.

Boot Strap

To make the program run a transputer, a `Bootstrap` must be added. This is a small piece of code which is added to the front of your code and contains instructions to reset the transputer and get ready to load and run your program. After the load has occurred the bootstrap is overlayed and disappears. To add this boot strap we use the `icollect` tool.

Run

To load the program onto the transputer we use the `iserver` program. This program sits on the host and enables the transputer to access the hosts services, these include the file system, keyboard and screen.

Debugging

If the program failed to run correctly then we enter the debugging phase of development. The toolset provides an interactive single stepping debugger called `idebug`. The user can specify breakpoints and trace through the code.

50.2.4 Software design cycle – multiple transputer systems

In the previous section we saw how to develop programs using single transputers, this section will concentrate on developing programs for *multi-transputer* systems. Diagram 50.2 shows the development cycle:

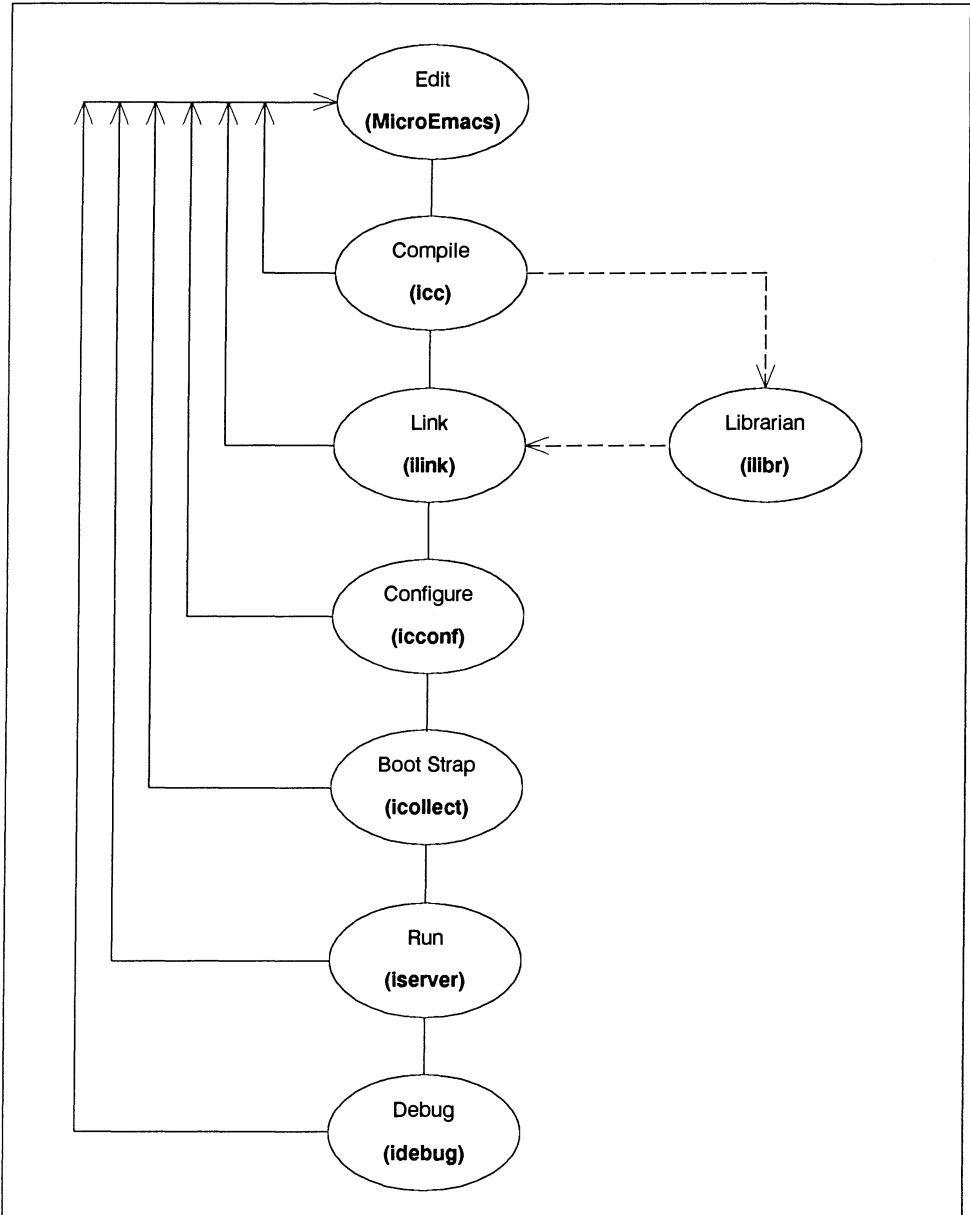


Figure 50.2 Typical software development route – multiple transputer systems

This development cycle is almost the same as for the single transputer systems, the exception is in the use of the configuration tool `icconf`.

Configure

Configuration is the step taken to map *processes* onto *processors*. The step consists of writing a *configuration script* which states how this mapping is to occur, we shall take a look at some example scripts later on. The tool for this step is called *icconf*.

50.3 Example problem description

50.3.1 Introduction

Throughout this document one simple example will be used. The system consists of the following components:

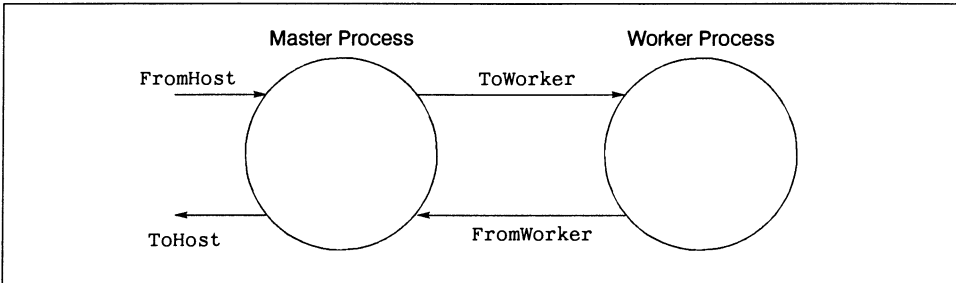


Figure 50.3 Example two process system

There are two parallel components, *Master* and *Worker*, these are called *processes*. They communicate with each other using **channels**, these are shown on the diagram as *ToWorker* and *FromWorker*. To access the host services, i.e. screen and keyboard, we use the *FromHost* and *ToHost* channels.

The *two processes* are written in C (we could have easily chosen another language e.g. Pascal, Fortran or Ada). The *Master* process takes input from the keyboard via *stdin* and passes it onto the *Worker* via the *ToWorker* channel. The *Worker* processes the data and passes it back to *Master* via the *FromWorker* channel. The *Worker* simply takes the keyboard characters and turns them into upper case characters, these are then displayed by the *Master* process.

50.3.2 What is configuration?

Configuration is the process by which individual components of the system are mapped onto physical processors. For the simple upper casing example we have two parallel communicating processes. This can be run as two parallel *processes* on one processor or as two processes running on two separate processors. Figure 50.4 shows how the *two processes* are mapped onto two separate processors.

Each transputer has *four* links, for this simple example we shall only use two links for communication, these are marked on the diagram as *LINK1* and *LINK2*. The *LINK0* is connected to the host machine, this is the link by which the system is booted and all communication with the host is passed back through this channel. For our example, all of the keys and output messages are passed via *LINK0*.

Each INMOS link is *bi-directional*, this means that you can map one input channel and one output channel onto *one* physical link.

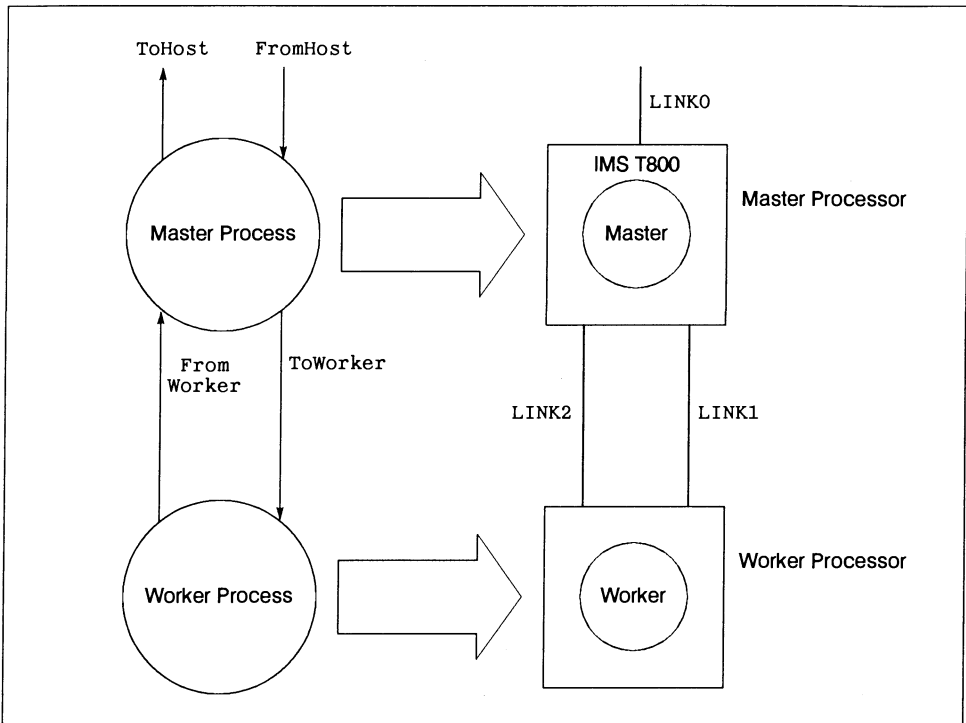


Figure 50.4 Example two processor system

50.4 Using the Dx214 C toolset

50.4.1 Introduction

In this chapter we shall use the **IMS Dx214 C Toolset** to build and configure the upper case example.

We shall use the example program in the following different ways:

- Upper casing on a single transputer, all in C.
- Upper casing on two transputers using the `icconf` configuration tool.

50.4.2 C parallel processing library extensions

As well as being an ANSI standard C compiler, `icc` provides a rich set of parallel processing library calls to enable us to write parallel programs *all* in C, for more in depth coverage of these routines the reader is referred to [2]. The extensions are similar to the ones provided in the **IMS D711 INMOS/3L Parallel C** compiler [1]. A table listing the features of the concurrency library is provided in section 50.2.

50.4.3 Parallel version on one transputer, all in C

In this section we shall look at how we use the C parallel library functions to create a parallel program running on a single transputer. The best way to understand how these libraries work is by the use of an example, so here goes:

```

/*
-----
-- MODULE: Upper Casing Example All in C, using icc.
--
-- FILE   : system.c
--
-- NAME   : Richard Onyett (Santa Clara, RTC)
--
-- PURPOSE:
--   To create and run two parallel processes to perform the
--   upper casing function.
-----
*/
#include <stdlib.h>
#include <stdio.h>
#include <channel.h>      /* New headers for the channel IO */
#include <process.h>     /* New headers for the processes */

/*
-- The Master and worker processes are in different files.
*/
extern void Worker (Process *p, Channel *FromMaster, Channel *ToMaster );
extern void Master (Process *p, Channel *FromWorker, Channel *ToWorker );

int main ( void )
{
    Process *WorkerPtr, *MasterPtr;      /* Declare some processes      */
    Channel *ToWorker, *FromWorker;     /* Connect 'em up with channels */

    printf ("Upper Casing EXAMPLE - STARTS\n");

    /*
    -- Allocate and initialise some channels.
    */

    if ( (ToWorker = ChanAlloc()) == NULL)
    {
        printf ("ERROR- Cannot allocate space for channel ToWorker\n");
        exit ( EXIT_FAILURE );
    }
    if ( (FromWorker = ChanAlloc()) == NULL)
    {
        printf ("ERROR- Cannot allocate space for channel FromWorker\n");
        exit ( EXIT_FAILURE );
    }

    /*
    -- Allocate the processes needed.
    */

    if ( (WorkerPtr =
        ProcAlloc ( Worker, 0, 2, ToWorker, FromWorker )) == NULL )
    {
        printf ("ERROR- Cannot allocate space for process Worker\n");
        exit ( EXIT_FAILURE );
    }
}

```

```

    }
    if ( (MasterPtr =
        ProcAlloc ( Master, 0, 2, FromWorker, ToWorker ) ) == NULL )
    {
        printf ("ERROR- Cannot allocate space for process MasterPtr\n");
        exit ( EXIT_FAILURE );
    }

    /*
    -- Now we have some processes, lets run 'em all in parallel...
    -- This will not return until ALL of the processes have finished..
    --
    */
    ProcPar ( MasterPtr, WorkerPtr, NULL);

    /*
    -- All processes have succesfully terminated, lets say so..
    */
    printf ("Upper Casing EXAMPLE - ENDS\n");

    exit ( EXIT_SUCCESS ); /* I'm Outta here */
}

```

Setting up a process

This example runs two processes `Master` and `Worker` in parallel. The `system.c` file contains the set up and calls to the two *processes*. Each *process* in the system is declared by saying:

```

#include <process.h>

Process *WorkerPtr;      /* Declare a process */

```

The *process* is defined as a function call i.e.

```

void Worker ( Process *Ptr, Param1, Param2, ... , ParamN )
{
    /* Do some things */
}

```

The `Process *` parameter *must* always be supplied, it is needed so that the process can be executed by the system.

To create the process we must use the `ProcAlloc` function i.e.

```

WorkerPtr = ProcAlloc ( Worker, 0, 2, ToWorker, FromWorker )

```

The parameters are as follows:

- 1 `Worker` The name of the function (process)
- 2 0 Default workspace size (4Kbyte on a 32-bit transputer and 1Kbyte on a 16-bit transputer).
- 3 2 Number of parameters to be passed, in this case its two.
- 4 `ToWorker` First parameter, in this case a channel called `ToWorker`.

5 FromWorker Second parameter, in this case a channel called FromWorker.

Note that WorkerPtr is the pointer to our process (or NULL if no space), ProcAlloc(Worker, 0, 2, ToWorker, FromWorker) is the function that builds processes.

Running the processes in parallel

Now that we have set up the processes we must run them in parallel. To achieve this we use the ProcPar library call i.e.

```
ProcPar ( MasterPtr, WorkerPtr, NULL );
```

The parameters to this are as follows:

- MasterPtr The master process.
- WorkerPtr The worker process.
- NULL No More processes.

Channel communications

To enable communication between our two processes we must declare some channels. This is done by saying:

```
Channel *ToWorker, *FromWorker;    /* Connect 'em up with channels */
```

We must now allocate some space for this channel by saying:

```
ToWorker = ChanAlloc();
```

This also initialises the channel.

The master process

The Master process takes a character from the standard input channel and passes it onto the worker process. The code for this process is as follows:

```

/*
-----
-- MODULE: Simple Parallel C example.
--
-- FILE   : master.c
--
-- NAME   : Richard Onyett (RTC, Santa Clara)
--
-- PURPOSE:
--   To generate a stream of ascii characters on a channel and pass them
--   to the upper case worker process.
--
-----
*/
#include <stdio.h>
#include <stdlib.h>
#include <channel.h>
#include <process.h>

/*
-- Declare a procedure called MASTER
*/
void Master (Process *p, Channel *FromWorker, Channel *ToWorker)
{
    int c;
    int Going = 1;

    p = p;                               /* Takes care of unused variable warning */

    printf ("Master C Process STARTING\n");

    while ( Going )
    {
        c = getchar();                    /* Get a character from the stdin      */
        ChanOutInt (ToWorker, c);        /* Pass to the worker                  */
        if (c == EOF)
            Going = 0;
        else
        {
            c = ChanInInt (FromWorker); /* Get the upper cased character back */
            putchar (c);                /* Output to the screen                */
        }
    }

    printf ("Master C Process ENDING \n");
}

```

This is the Master procedure. A character is read in and passed along a channel to the worker process. Channel communication is done using the ChanOutInt and ChanInInt library calls.

The worker process

The worker process takes a character from an input channel and converts it to upper case, it then passes the new character back along an output channel to the master process. The code for the worker process is as follows:

```
/*
-----
-- MODULE: Simple Parallel C example.
--
-- FILE   : worker.c
--
-- NAME   : Richard Onyett (RTC, Santa Clara)
--
-- PURPOSE:
--   To receive a stream of ascii characters on a channel and convert them
--   to upper case, whether they want to be or not!!
--
-----
*/
#include <ctype.h>
#include <process.h>
#include <channel.h>

/*
-- Declare the worker process.
*/
void Worker ( Process *p, Channel *FromMaster, Channel *ToMaster )
{
    int key = 0;
    int Going = 1;

    p = p;                                     /* Takes care of unused variable warning */

    while ( Going )
    {
        key = ChanInInt ( FromMaster );        /* Get a key from the MASTER */
        if (key == EOF)
            Going = 0;
        else
            ChanOutInt (ToMaster , toupper(key) ); /* Pass it back          */
    }
}
```

Again, channel communication is done using the ChanOutInt and ChanInInt library routines.

Building the upper case example

The C components are built using the following makefile

```
#
# Module: Simple Parallel C Examples
# Name : Richard Onyett (Santa Clara, RTC)
# File : makefile
#
CC = icc
CFLAGS = /t8
SRC = master.tco worker.tco system.tco
LINK = ilink /f startup.lnk $(SRC) /t8
BOOT = icollect system.lku /t

system.btl: $(SRC)
            $(LINK) /o system.lku
            $(BOOT)

system.tco:    system.c
              $(CC) system.c $(CFLAGS)

master.tco:    master.c
              $(CC) master.c $(CFLAGS)

worker.tco:    worker.c
              $(CC) worker.c $(CFLAGS)
```

To compile we say

```
$ make
```

This will invoke the C compiler (icc) and the linker (ilink), the output of the linker is then used to produce a bootable file which can be downloaded to the transputer.

Running the single processor version

To run the program we type:

```
$ iserver /se/sb system.btl
Booting Root Transputer.....
Upper Casing EXAMPLE - STARTS
Master C Process STARTING
a
A
hello
HELLO
^Z
Master C Process ENDING
Upper Casing EXAMPLE - ENDS
```

In detail,

- iserver Invoke the host server program.
- /se/sb system.btl Boot the file system.btl to the transputer and monitor the error flag.
- Booting Root Transputer..... Message from the server as it starts.

50.4.4 Configuring a multi-processor version using icconf

Introduction

Now that we have the two process version working we shall map this onto two transputers using the **configurer**. The **IMS Dx214 Toolset** provides a *C like* configuration language to specify how the mapping is to occur. The configurer tool is called `icconf`, which stands for **Inmos C Configurer**. The physical system was shown earlier in figure 50.3. The reader should refer to this diagram as it will make this section clearer.

As before, intimate details of the configuration language syntax is ignored, the interested reader is referred to the **IMS Dx214 ANSI C Toolset User Manual** [2].

Introducing some new tools

In section 50.2.4 we saw how to use the tools to develop a multi transputer program. The main difference between this development and the development for single processors is the use of the configuration tools to map the *processes* onto *processors*. The following diagram shows the steps presented in section 50.2.4, but shows the filename conventions used:

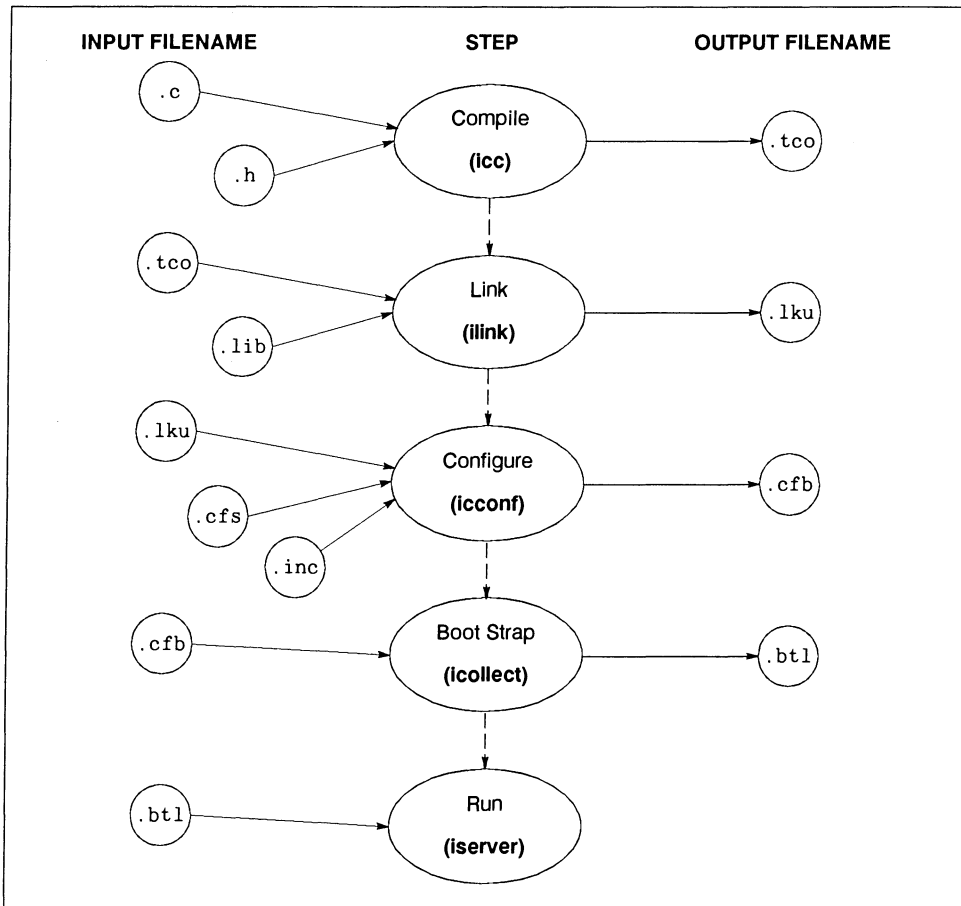


Figure 50.5 Filename conventions for multiple transputer development

The `icconf` tool takes, as input, a textual description of the transputer network, with a file suffix of `.cfs`, and outputs a configuration data file, with a `.cfb` suffix. This file is then passed into a code collector tool called `icollect`. The output from this tool is our bootable file that we can run on our network of transputers, using the `iserver`. The `.inc` files are predefined descriptions of transputer modules (TRAMs).

The master process

To enable the processes to be configured onto processors, we must add some code to form a process interface. Each process must have a `main` wrapped around it. The C code for the Master process is as follows:

```

/*
-----
-- MODULE: Simple Parallel C example.
--
-- FILE   : master.c
--
-- NAME   : Richard Onyett (RTC, Santa Clara)
--
-- PURPOSE:
--   To generate a stream of ascii characters on a channel and pass them
--   to the upper case worker process.
-----
*/
#include <stdlib.h>
#include <stdio.h>
#include <misc.h>
#include <channel.h>
#include <ctype.h>

int main ( void )
{
    Channel *ToWorker;
    Channel *FromWorker;

    int c;
    int Going = 1;

    /*
    -- Access the external configuration channels.
    */
    FromWorker = (Channel *) get_param (3);
    ToWorker   = (Channel *) get_param (4);

    /*
    -- Intro
    */
    printf ("Enter some text and it will be upper cased      \n" );
    printf ("\nTo quit type EOF (CTRL-Z)                    \n" );

    while ( Going )
        {
            c = getchar();
            ChanOutInt ( ToWorker, c );

            if ( c == EOF)
                Going = 0;
            else
                {
                    c = ChanInInt ( FromWorker );
                }
        }
}

```



```

        putchar (c);
    }
}

exit_terminate ( EXIT_SUCCESS ); /* Shake the needles from your back */
}

```

The worker process

The code for the worker process is as follows:

```

/*
-----
-- MODULE: Simple Parallel C example.
--
-- FILE   : worker.c
--
-- NAME   : Richard Onyett (RTC, Santa Clara)
--
-- PURPOSE:
--   To receive a stream of ascii characters on a channel and convert them
--   to upper case, whether they want to be or not!!
-----
*/
#include <stdlib.h>
#include <misc.h>
#include <ctype.h>
#include <channel.h>

int main ( void )
{
    int key = 0;
    int Going = 1;

    Channel *FromMaster;
    Channel *ToMaster ;

    FromMaster = (Channel *) get_param (1);
    ToMaster   = (Channel *) get_param (2);

    while ( Going )
    {
        key = ChanInInt ( FromMaster );

        if (key == EOF)
            Going = 0;
        else
            ChanOutInt ( ToMaster, toupper(key) );
    }
}

```

Notice the use of the `get_param` function to connect to the external configuration channels.

Connecting to the external configuration channels

Using the **IMS D711** configuration language, we have used the following code to connect to an external configuration channel (the Dx214 still supports this method for compatibility):

```

#define OUT_CHAN 2
#define IN_CHAN 2

```

```

int main (int argc, char* argv[], char* envp[],
          Channel* in[], int inlen, Channel* out[], int outlen)
{
    int c;

    /* Put out the official number on channel 2 */
    ChanOutInt ( out[OUT_CHAN], 38);

    ... other statements

    /* Pull in a INT on channel 2 */
    c = ChanInInt ( in[IN_CHAN ]);
}

```

For icconf we use the `get_param(param_number)` function call. This returns a pointer to the external configuration channel connected to `param_number`, we shall see later on how we obtain the value for `param_number`. So to obtain a connection to the outside world we write:

```

Channel *ToWorker; /* Declare a local channel */

/*
-- Connect to the external config channel
*/
ToWorker = (Channel *) get_param (4);

```

It is worth mentioning that the configuration channels zero and one are reserved for by the C run-time library.

Writing a configuration script

Now we are in a position to look at the configuration script required to build the example on two processors.

The configuration script is held in a file called `upc.cfg`

```

/*
-- MODULE: Configuration script file for a simple two processor program
--
-- NAME : Richard Onyett (RTC, Santa Clara)
--
-- PURPOSE:
-- To configure (using INMOS C configuration language) the UPPER CASE
-- program.
*/

/*
-- Declare all of the physical hardware in the system. We have
-- 2 x T800 TRAMs in ours.
*/
T800 (memory = 2M) root ;
T800 (memory = 2M) Slave;

/*
-- Tell the world how these two are intimately bonded
*/
connect root.link[0], host;
connect root.link[3], Slave.link[0];

/*
-- Describe the interface between our processes

```

```

*/
/*
-- Master task
--
--           fs -----> |         | -----> ToWorker
--           ts <----- |         | <----- FromWorker
--           -----
*/
process (stacksize=1k, heapsize=50k,
        interface ( input fs, output ts, input FromWorker, output ToWorker ) ) Master;

/*
-- Worker/Slave task
--
--   FromMaster -----> |         |
--   ToMaster   <----- |         |
--   -----
*/
process (stacksize=1k, heapsize=50k,
        interface ( input FromMaster, output ToMaster ) ) Worker;

/*
--
-- Describe how things are wired up
--
*/
input from_host;
output to_host ;

connect Master.fs, from_host;
connect Master.ts, to_host;
connect Master.ToWorker, Worker.FromMaster;
connect Master.FromWorker, Worker.ToMaster ;

/*
--
-- Pull in the "real code"
*/
use "master.lku" for Master;
use "worker.lku" for Worker;

/*
--
-- Place the processES onto processors
*/
place Master on root;           /* Run Master on the first transputer */
place Worker on Slave;         /* Run Worker on the second transputer */

place from_host on host;       /* Wire up to the HOST machine */
place to_host on host;

place Master.fs on root.link[0];
place Master.ts on root.link[0];

/*

```

```
--
-- Configured for IMS B008
--
*/
place Master.ToWorker   on root.link[3];
place Master.FromWorker on root.link[3];

place Worker.ToMaster   on Slave.link[0];
place Worker.FromMaster on Slave.link[0];
```

Declaring the physical hardware

We must declare all the transputers nodes in the network.

In detail,

- T800 Type of processor.
- (memory = 2M) Describe amount of memory available on this processor.
- root; Name of the transputer node.

Showing physical interconnect

We must describe how the links of our root and Slave processors are connected. The root node is connected by its Link 0 to the host computer.

In detail,

- connect root.link[0], host; Sign up for the host services.
- connect root.link[3], Slave.link[0]; Connect root node to Slave node.

Interface description

This section describes how our process is joined to all others in the system. We pass in channel parameters and data concerning the size of the stack and heap required.

In detail,

- process Configuration keyword.
- stacksize=1k Size of the stackspace.
- heapsize=50k Size of the heap.
- interface Configuration keyword. Here comes the interface description.
- input fs Pass input channel called fs.(Channel parameter 1).
- output ts Pass input channel called ts.(Channel parameter 2).

(The first two items in the interface description for a process communicating with the host must be the from server (fs) and to server (ts) channels, in that order.)

- input FromWorker Pass input channel called FromWorker.(Channel parameter 3).
- output ToWorker Pass output channel called ToWorker.(Channel parameter 4).
- Master Name of this process.

This interface section is how channel parameters are passed into the C processes (although the parameters do not have to be channels). The textual declaration of the channels determines which number should be used in the get_param call.

WARNING! There is no checking that the parameter number is wired to the correct channel.

Wiring things up

The next section wires up the soft channels to the processes on the transputer nodes.

In detail,

- `connect Master.fs, from_host`; Connect to the server on the host.
- `connect Master.ts, to_host`; Connect to the server on the host.
- `connect Master.ToWorker, Worker.FromMaster`; Connect the Master to the Worker.
- `connect Master.FromWorker, Worker.ToMaster` ;

Pulling in the real code

To pull in the actual compiled and linked code we use the use directive.

- `use "master.lku" for Master`; Pull in the master process.
- `use "worker.lku" for Worker`; Pull in the worker process.

Placing the processes onto processors

The final stage is to place all of our processors onto a transputer node and *place* the soft channels onto physical transputer links.

- `place Master on root`; Run Master on the first transputer.
- `place Worker on Slave`; Run Worker on the second transputer.
- `place Master.fs on root.link[0]`; Connect up the server channels.
- `place Master.ts on root.link[0]`; Connect up the server channels.
- `place Master.ToWorker on root.link[2]`; Connect up the Master using channel 2.
- `place Master.FromWorker on root.link[2]`;
- `place Worker.ToMaster on Slave.link[1]`; Connect up the Worker using channel 1.
- `place Worker.FromMaster on Slave.link[0]`;

Building the example

Now we have configuration script written we must build the whole lot! To do this we use the following make-file

```
#
# MAKEFILE to build simple C program.
#
E = .bt1
O = .lku
CC = icc
CFLAGS = /t8
OBSJ = master.tco worker.tco
LINK1 = ilink /f startup.lnk master.tco /t8 /o master.lku
LINK2 = ilink /f startrd.lnk worker.tco /t8 /o worker.lku
CONFIG = icconf upc.cfg
BOOT = icollect upc.cfb
```

```
upc$(E):      $(OBJS) master.lku worker.lku
              $(CONFIG)
              $(BOOT)

master.lku:   master.tco
              $(LINK1)

worker.lku:  worker.tco
              $(LINK2)

master.tco:  master.c
              $(CC) master.c $(CFLAGS)

worker.tco:  worker.c
              $(CC) worker.c $(CFLAGS)
```

The configurer is run by

```
$ icconf upc.cfs
```

In detail,

- `icconf` Invoke the configurer.
- `upc.cfs` The name of the configuration script.

The final stage, to collect all of the code, is done by

```
$ icollect upc.lku
```

Running the multi-processor version

This is done exactly as described previously on page 448.

50.5 Conclusions

This document has taken a look at the various methods for writing parallel C programs. The user has the choice between a program written and configured *completely* in C or using a **Mixed language** approach by wrapping the C programs in occam. All of the methods allow the user to initially test the system on a single transputer, once this is working then a configuration script is written and the system is parcelled out onto multiple transputers.

50.6 Differences between 3L and icc concurrency library

Table 50.2 provides a comparison between the IMS D711 and IMS D714 concurrency libraries.

icc	3L	icc	3L
channel.h	chan.h	ProcGetPriority	thread_priority
Channel	CHAN	PROC_HIGH	THREAD_URGENT
ChanAlloc	Not available	PROC_LOW	THREAD_NOTURG
ChanInit	chan_init	process.h	timer.h
ChanReset	chan_reset	ProcAfter	timer_delay
ChanIn	chan_in_message	ProcWait	timer_wait
ChanInChar	chan_in_byte	ProcTime	timer_now
ChanInInt	chan_in_word	ProcTimeAfter	timer_after
ChanInTimeFail	chan_in_message_t	ProcTimePlus	Not available
ChanInChanFail	Not Available	ProcTimeMinus	Not available
ChanOut	chan_out_message	semaphor.h	sema.h
ChanOutChar	chan_out_byte	SemAlloc	sema_alloc
ChanOutInt	chan_out_word	SemInit	sema_init
ChanOutTimeFail	chan_out_message_t	SemWait	sema_wait
ChanOutChanFail	Not Available	SemSignal	sema_signal
process.h	thread.h	SEMAPHOREINIT	Not Available
Process	Not Available	Not available	sema_signal_n
ProcRun	thread_create	Not available	sema_wait_n
ProcPar	Not Available	Not available	net.h
ProcParList	Not Available	Not available	net_send
ProcAlloc	thread_create	Not available	net_receive
Proclnit	thread_create	Not available	par.h
ProcRunHigh	thread_create	printf	par_printf
ProcRunLow	thread_create	free	par_free
ProcReschedule	Not Available	malloc	par_malloc
ProcParam	Not Available	fprintf	par_fprintf
ProcStop	thread_stop		

Table 50.2 Comparison of icc vs 3L concurrency library functions

Upgrading code to IMS Dx214 from IMS D711

If you are recompiling code from **IMS D711** with the **IMS Dx214 C** toolset you should include the `conndx11.h` header file. This contains macros to convert the **IMS D711** channel and thread calls to equivalent **IMS Dx214** calls.

50.7 Useful hints when writing IMS Dx214 C Toolset programs

Here are a few simple rules that you may find useful when writing parallel programs using the IMS Dx214 C Toolset.

Program design methodology

Writing parallel programs can be a confusing task! Good program design is *essential* when writing software that may execute on tens or even hundreds of separate processors. Further details on writing concurrent software are available in [3].

Let us assume that you have broken the problem down into parallel blocks, an easy way to test the system is to write a *software simulation* model. This means running many parallel processes on a single transputer. We have done this already with our upper case example (see section 50.4), where we used the ProcPar command to run the Master and Worker processes on a single transputer. Once we have this system working we simply write a configuration script and use that to map the processes onto processors. Always have your processes defined in separate files! This means that you can reuse the code in both a simulation and configured modes. For each of your processes add the following:

```

/*
-- Master process
*/

#ifdef CONFIG
int main ( void )
{
    Channel *ToWorker;
    Channel *FromWorker;

    /*
    -- Access the external configuration channels.
    */
    FromWorker = (Channel *) get_param (3);
    ToWorker   = (Channel *) get_param (4);
#else
void Master ( Process *Mptr, Channel *FromWorker, Channel *ToWorker )
{
#endif

    ... Rest of the process
}

```

This code segment makes use of a conditional compilation flag called CONFIG, when this is set it will turn the process into one which can be configured. If the flag is not set then the process is assumed to be running on one processor.

50.7.1 What if the program will not run?

Now that you have compiled, linked, collected, configured and submitted your program to a transputer (using iserver), you discover that it will not function correctly. This usually means that the program runs for a time then simply grinds to a halt, the following is a checklist you may find useful:

- Make sure that any external transputers, such as in an INMOS ITEM rack, have power!
- Use the checkout tools to see if your transputer network is what you thought it was!
- If the program requires setting of any IMS C004s, has this been done?
- Compile with the debugging option (/g) on and use the debugger!

- Run the `iserver` with the `/se` option on to test if the error flag has been set.
- Ensure that all `ProcPar` calls are NULL terminated.
- Ensure that all `ProcAlt` calls are NULL terminated.
- When you specify that there are `n` parameters passed to a *process* make sure all `n` parameters are passed.
- Make sure you have used `exit_terminate` for configured programs. The server will not terminate until this has been executed!
- Make sure you have allocated some space for a channel.
- Check that the `get_params` are connected to the correct channels in the configuration script.
- Check that the transputers, in the network, have enough memory. Do not specify 1 MByte in the configuration script when only 32K is available on the `TRAM`.
- Does the processor *type* in the network match what you have compiled for?
- Make sure that there is enough stack and heap space.

50.8 References

- 1 IMS D711 3L Parallel C user manual, INMOS Limited, February 1989 (INMOS document number 72 TDS 179 00)
- 2 IMS Dx214 ANSI C toolset user manual, INMOS Limited, August 1990 (INMOS document number 72 TDS 224 00)
- 3 *Program design for concurrent systems*, INMOS Technical Note 5, Philip Mattos, INMOS Limited, Bristol.



Appendices

A Quality and Reliability

Systems products are embraced within the INMOS Quality Policy which incorporates specific programmes in the following areas:

- Design in quality
- New product verification phase
- Document Control
- Quality control monitors
- Production soak testing
- Reliability testing
- Software engineering standards

All systems products are designed in house using CAD facilities specific to PCB manufacture. These facilities incorporate design simulation and provide production data which helps to reduce design to production problems.

During the product verification phase, the new product is evaluated and its build/test specifications are endorsed.

All system products are assembled and tested at INMOS approved assembly houses which conform to the INMOS Quality Program. Quality procedures detail the build specification, production testing and final product status. These procedures are all monitored and controlled by the Document Control Department (DCD).

In circuit automatic testing provides an effective monitor to the production phase by providing assembly and component analysis.

An INMOS Quality Assurance sample test evaluates all production batches. At this stage the conformance of the product is confirmed and the test data logged for reference.

Reliability testing is carried out on the major product lines. Samples are taken from standard stock and subjected to life testing.

Software is produced to INMOS software engineering standards, which encompass design methods, design verification, configuration management, coding practices, inspection and test procedures, and build and release controls.

Production media is manufactured by INMOS approved copy houses, and an INMOS Software Quality Assurance sample test evaluates all production batches.

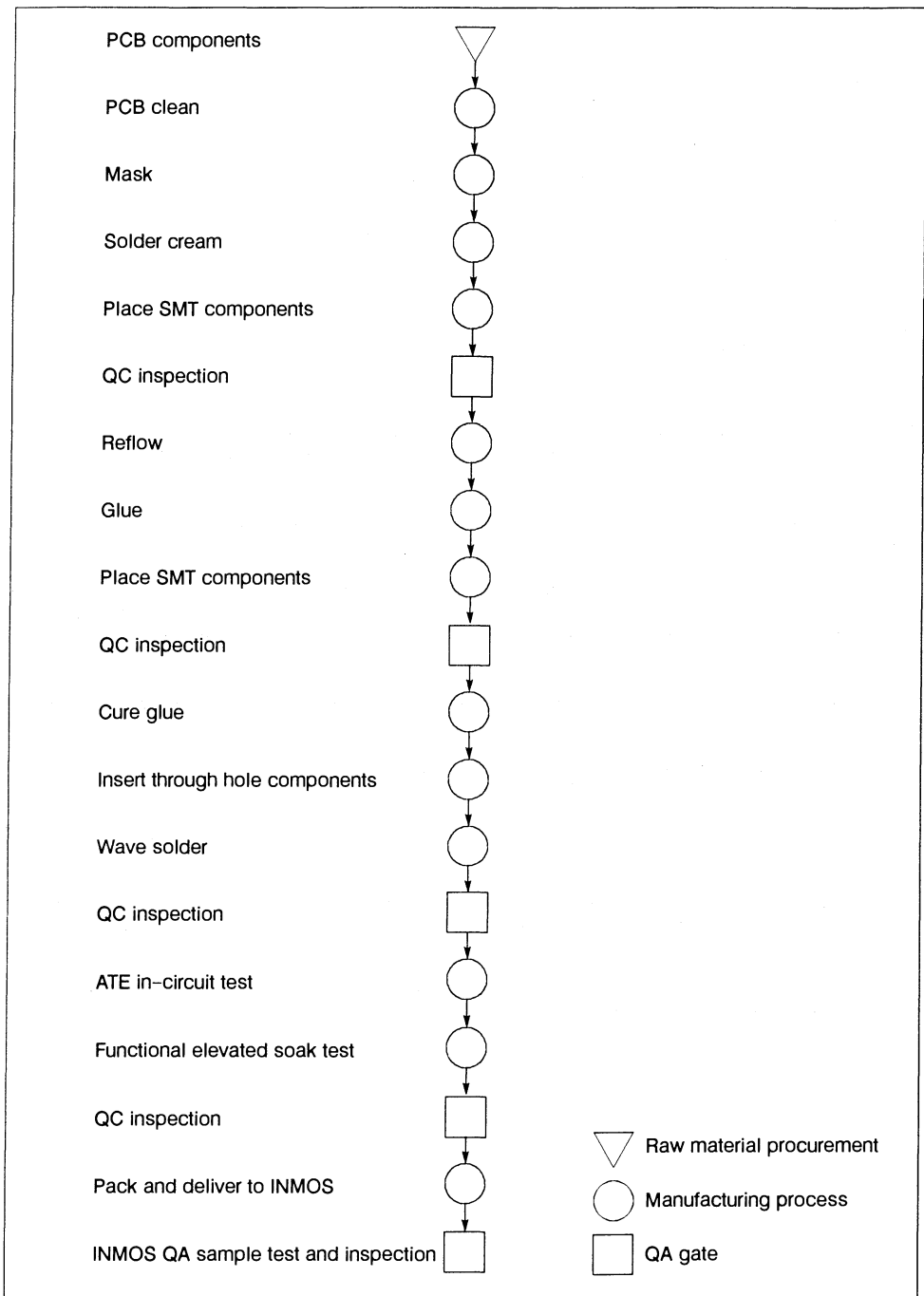


Figure A.1 Product Flow

B Software Licensing

End User Licences

All INMOS standard software products (development tools and board support packages) are supplied with a 'shrink-wrapped' licence. By breaking the seal, the user accepts the specified Terms and Conditions.

Distribution Licences

It is the strategic aim of INMOS to make the transputer development tools available on a wider range of host machines.

With this in mind INMOS offers two solutions for third parties to support transputer development tools on new host machines.

'Transputer hosted' operation can be supported by the provision of a hardware interface, the writing of a device driver and the porting of a server program. Binary versions of the development tools can then be executed on the attached transputer hardware. INMOS is able to support third parties in this development by providing

- The source code of the server program
- The source code of example device drivers
- Volume purchase agreements for copies of the standard development tool packages

Alternatively, INMOS may also support third parties who wish to port the standard development tools, working in 'cross-development' mode, to new computer platforms. This involves the release of the source code of the development tools themselves, and will only be considered if the third party can be seen to satisfy stringent requirements concerning the technical support and marketing of the resultant product(s).

Appendix	Page
A	471
B	472
C	473
D	474
E	475
F	476
G	477
H	478
I	479
J	480
K	481
L	482
M	483
N	484
O	485
P	486
Q	487
R	488
S	489
T	490
U	491
V	492
W	493
X	494
Y	495
Z	496

C Product Reference Tables

C.1 Table of Composite Products

Product Code	Product Description	Constituent Parts
IMS B008-1	IBM PC Motherboard	IMS B008 IBM PC Motherboard IMS S708 Motherboard Support pack
IMS B012-1	Eurocard Motherboard	IMS B012 Eurocard Motherboard IMS L012A Motherboard Support Pack IMS S006A Support software
IMS B017-1	IBM PS/2 Motherboard	IMS B017 PS/2 board IMS S217A Support software
IMS B419-4	G300 Graphics TRAM	IMS B419 Graphics TRAM IMS CA13 cable IMS F003A 2D Graphics library support software
IMS B420-3 IMS B420-5	Vector Processing TRAM	IMS B420 Vector Processing TRAM IMS F000A VecTRAM library support software
IMS B421-10	GPIB TRAM	IMS B421 GPIB TRAM F001A GPIB library support software
IMS B422-10	SCSI TRAM	IMS B422 SCSI TRAM IMS F002A SCSI library support software

C.2 Table of Board Products and Associated Software

Product Code	Product Description	Associated Software	Product included with
IMS B419	G300 Graphics TRAM	IMS F003	IMS B419-4
IMS B420	Vector processing TRAM	IMS F000	IMS B420-3 IMS B420-5
		IMS F007 VecTRAM libraries and development tools	*
IMS B421	GPIB TRAM	IMS F001	IMS B421-10
IMS B422	SCSI TRAM	IMS F002	IMS B422-10
IMS B429	Video processing TRAM	IMS F004	IMS B429
IMS B008	PC Motherboard	IMS S308, IMS S708	IMSB008-1
IMS B014	VMEbus slave board	IMS S514	*
IMS B016	VMEbus master/slave board	IMS S514	*
IMS B015	NEC PC Motherboard	IMS S308, IMS S708	IMS B015-1
IMS B017	IBM PS/2 Motherboard	IMS S217	IMS B017-1

* Available as a separate product



Indexes

Index by product number

For the purposes of this index the product number prefix 'IMS' is not shown.

B008 309	CA12 383
B012 361	CA13 385
B014 329	CA14 385
B015 353	
B016 333	D4205 9
B017 321	D4214 23
B018 375	D4216 43
B250 381	D4217 35
B300 377	D5205 9
B401 149	D5214 23
B404 173	D5216 43
B408 209	D5217 35
B409 219	D6205 9
B410 161	D6214 23
B411 167	D6216 43
B415 229	D6217 35
B416 155	D7205 9
B417 189	D7214 23
B418 237	D7216 43
B419 243	D7217 35
B420 255	
B421 265	
B422 283	F000B 69
B426 197	F001B 83
B427 203	F002B 93
B428 181	F003A 109
B429 295	F007A 119
B430 301	
B431 305	S217 127
	S308 127
CA01 385	S507 131
CA03 385	S514 127
CA09 385	S607 131
CA10 385	S707 131
CA11 385	S708 127

Index by product name

- 32Kbyte TRAM, 149
- 64Kbyte TRAM, 155
- 160Kbyte TRAM, 161
- 1Mbyte TRAM, 167
- 2Mbyte TRAM
 - IMS T801 transputer, 173
 - IMS T800 transputer, 181
- 4Mbyte TRAM
 - Size 1, 197
 - Size 4, 189
- 8Mbyte TRAM, 203
- 2D graphics libraries, 109

- Ada compiler, 53
- ALSYS Ada compiler, 53
- ANSI C toolset, 23
- ANSI FORTRAN 77 toolset, 43

- C executive, 141
- C program development, 435
- C + +, 35
- Cables for board products, 385
- Card frame adaptor, 383

- Development kits, 63
- Device driver
 - & motherboard support software, 127
- Differential link TRAM, 229
- Display TRAM, 219
- Double extended eurocard, 361
- Dual inline transputer modules, 389

- Ethernet TRAM, 305

- Flash ROM TRAM, 237
- Frame store TRAM, 209

- Glockenspiel C + +, 35
- GPIB libraries, 83
- GPIB TRAM, 265
- Graphics libraries, 109

- IBM PC motherboard, 309
- IBM PS/2 motherboard, 321
- IEEE-488 GPIB TRAM, 265
- Integrated graphics TRAM, 243

- Libraries
 - 2D graphics, 109
 - GPIB, 83
 - SCSI, 93
 - VecTRAM, 69, 119
- Licensing, software, 465

- Module motherboard architecture, 409
- Motherboard support software, 127

- NEC 9800 series PC board, 353
- Network connection device, 377
- Network support software, 131

- OCCAM 2 toolset, 9

- Parallel C, program development, 435
- Product reference tables, 467
- Prototyping TRAM, 301

- Quality and reliability, 463

- Real-time executive, 139

- SCSI libraries, 93
- SCSI TRAM, 283
- Software licensing, 465

- TRAM motherboard, 375
- TRAMs, 389

Transputer development kits, 63

Transputer modules, 389

Vector processing TRAM, 255

VecTRAM libraries,

 C interface generator & linker, 119

VecTRAM library, 69

Versatile real-time executive, 139

Video Image Processing TRAM, 295

VME rack, 381

VMEbus master/slave board, 333

VMEbus slave card, 329

VRTX32/T real-time executive, 139